TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

SAMI HARA

MALLIPOHJAINEN SYSTEEMISUUNNITTELU AUTOMAATION OPPIMISYMPÄRISTÖN UUDISTUKSESSA

Diplomityö

Tarkastaja: professori Matti Vilkko
Tarkastaja ja aihe hyväksytty
Teknisten tieteiden tiedekuntaneuvoston kokouksessa 8. lokakuuta 2014

TIIVISTELMÄ

SAMI HARA: Mallipohjainen systeemisuunnittelu automaation oppimisympäristön uudistuksessa
Tampereen teknillinen yliopisto
Diplomityö, 103 sivua
Huhtikuu 2015
Automaatiotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Systeemiteoria
Tarkastaja: professori Matti Vilkko

Avainsanat: systeemisuunnittelu, mallipohjainen systeemisuunnittelu, SysML, oppimisympäristö, automaatio, prosessien hallinta

Realistiset oppimisympäristöt toimivat oppimisen tukena. Niiden avulla voidaan selventää teorian soveltamista käytännön tilanteissa. Tässä diplomityössä on tutkittu, miten systeemisuunnittelun ja mallipohjaisen systeemisuunnittelun menetelmiä voidaan käyttää hyväksi monikäyttöisen automaation ja prosessien hallinnan oppimisympäristön suunnittelussa.

Työssä on esitelty systeemisuunnittelun perusperiaatteet sekä mallipohjaisen systeemisuunnittelun menetelmiä ja sovelluksia. Tutkimuksen kohteena olevan oppimisympäristön yksi merkittävin osa on sen automaatiojärjestelmä. Tämän myötä diplomityön yksi luku käsittelee automaatiojärjestelmien yleisimpiä teknologioita.

Systeemin mallin luomisessa käytettiin apuna mallipohjaisen systeemisuunnittelun OO-SEM-menetelmää sekä SysML-mallinnuskieltä. Kyseinen mallinnuskieli koostuu yhdeksästä erilaisesta kaaviosta, joiden avulla voidaan mallintaa systeemin rakennetta, rakenneosien välistä vuorovaikutusta sekä käyttäytymistä.

Olemassa olevan oppimisympäristön ominaisuudet selvitettiin ja määritettiin sidosryhmien vaatimusten perusteella muutokset ja hankinnat. Lisäksi tehtiin suunnitelmat muutoksista, joita oppimisympäristöön täytyy tehdä tulevaisuudessa. Oppimisympäristöstä pyrittiin kehittämään moderni ja monikäyttöinen, joka tukee automaation ja prosessien hallinnan opetusta sekä mahdollista tutkimusta.

Työssä tehty systeemin malli tukee oppimisympäristön määrittelyssä, suunnittelussa, asennuksissa ja ylläpidossa. Mallipohjaisten menetelmien avulla pystyttiin varmistamaan, että eri sidosryhmien alkuperäiset vaatimukset täyttyvät. Tärkeimpinä asioina nousi esiin vaatimusten mallintaminen ja jäljitettävyyden ylläpitäminen.

ABSTRACT

SAMI HARA: Redeveloping the learning environment for automation by model-based systems engineering
Tampere University of Technology
Master of Science Thesis, 103 pages
April 2015
Master's Degree Programme in Automation Technology
Major: Systems theory
Examiner: Professor Matti Vilkko

Keywords: systems engineering, model-based systems engineering, SysML, learning environment, automation, process control

Realistic learning environments are great way for supporting the learning process. They can be used to clarify how to apply theory in practice. This thesis studies how systems engineering and model-based systems engineering methods can be used in the design of multi-purpose learning environment for automation and process control.

In this thesis the basics of the systems engineering and some model-based systems engineering (*MBSE*) methods are introduced. Process control system is one of the key components of the learning environment. Therefore the trends of the automation and process control systems are also studied.

The system model used in the design was created with model-based systems engineering method OOSEM and Systems Modeling Language (*SysML*). SysML consists of nine different diagrams which can be used to model structure, interconnection and behavior of systems.

The legacy system of the learning environment was studied. Defined purchases and plans for the future were based on that study and stakeholder requirements. The learning environment was developed to be modern and multi-purpose that supports teaching and research of automation and process control.

The system model supports definition, design, installation and maintenance of the learning environment. Methods used in the thesis proved to be useful in the design of redeveloped learning environment. Especially requirements modeling and establishing traceability turned out to be significant.

ALKUSANAT

Tämä diplomityö on tehty Tampereen teknillisen yliopiston systeemitekniikan laitokselle oppimisympäristöjen kehittämiseen liittyen. Työ on rahoitettu TTY:n tukisäätiön apurahalla.

Haluan kiittää työn tarkastajaa professori Matti Vilkkoa mielenkiintoisesta aiheesta sekä osaavasta ohjauksesta työhön ja kirjoitusprosessiin liittyen. Lisäksi haluan kiittää Sami Rajalaa ja Annimaria Virtasta, jotka työskentelivät osaksi saman projektin parissa ja toivat näkökulmia tämänkin työn osaksi. Lopuksi suuri kiitos vanhemmilleni, jotka ovat kannustaneet ja tukeneet koko opintojeni ajan.

Tampereella, 24.3.2015

Sami Hara

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

ACN MR/SR1	Application and Control Node Medium Rail/Small Rail mounted. Metson prosessiasematyyppejä.
Booch	Menetelmä oliopohjaiseen ohjelmistokehitykseen.
DCOM	Distributed Component Object Model. Microsoftin kehittämä menetelmä tiedonsiirtoon erilaisten ohjelmistojen välillä.
DCS	Distributed control system. Hajautettu ohjausjärjestelmä.
EAS	Engineering Application Station. Metson suunnittelupalvelin.
FF	Foundation Fieldbus. Eräs kenttäväylästandardi.
HART	Highway addressable remote transducer protocol. Digitaalinen kommunikointiprotokolla virtaviestisignaalin johtimissa.
IDEF0	ICAM Definition for function modeling. Systeemisuunnittelussa käytettävä toiminnallisen mallintamisen menetelmä.
INCOSE	International council of systems engineering. Kansainvälinen systeemisuunnittelujärjestö.
MARTE	Modeling and Analysis of Real Time and embedded systems. Reaaliaikajärjestelmien suunnitteluun tarkoitettu mallinnuskieli.
MBP	Manchester encoded Bus Powered. Tietoliikenteessä käytetty tiedon koodausmentelmä. Mahdollistaa myös tehonsiirron väylässä.
MBSE	Model-based Systems Engineering. Mallipohjainen systeemisuunnittelu.
MDA	Model Driven Architecture. Ohjelmistotuotannon menetelmä mallien alustariippumattomuuteen.
MOE	Measures of effectiveness. Vaikuttavuuden mittari. Systeemisuunnittelun apuna käytettävä tekninen mittari.
MOF	Meta-Object Facility. Mallien elementtien hallintaan tarkoitettu kehys.
MOP	Measures of performance. Suorituskyvyn mittari. Systeemisuunnittelun apuna käytettävä tekninen mittari.
OLE	Object Linking and Embedding.
OMG	Object Management Group. Ohjelmistotekniikan standardeja ylläpitämä yhteenliittymä.
OMT	Object-modeling technique. Mallipohjainen menetelmä oliopohjaiseen ohjelmistokehitykseen.
OOSE	Object-oriented software engineering. Oliophjainen ohjelmointikieli ja menetelmä.
OOSEM	Object-Oriented Systems Engineering Method. ISO/IEC 15288 standardiin perustuva mallipohjaisen systeemisuunnittelun menetelmä.
OPC	OLE for Process Control. Automaatiotekniikan standardi tiedonsiirtoon ohjausjärjestelmien ja tietojärjestelmien välillä.
OSI	Open System Interconnection. Tietoliikenteen käsitemalli.
PHOY	Työssä suunnittelussa apuna käytetty lyhenne prosessien hallinnan oppimisympäristöstä.
PLC	Programmable Logic Controller. Ohjelmoitava logiikka
Profibus	Process Field Bus. Kenttäväylästandardi automaatiotekniikassa.
Profibus DP	Decentralized Peripherals. Käytetään operoimaan sensoreita keskuskontrollerin avulla tehdasautomaatiosovelluksissa

Profibus PA	Process automation. Profibus tekniikan variaatio, joka on tarkoitettu etenkin prosessiautomaation sovelluksiin.
SE	Systems engineering. Systeemisuunnittelu. Suurten ja monimutkaisten systeemien suunnitteluun käytettävä systeemin koko elinkaaren huomioonottava kokonaisvaltainen lähestymistapa.
SysML	Systems modeling language. Systeemisuunnittelun tarpeisiin kehitetty mallinnuskieli. UML kielen profiili.
SYSMOD	Systems Modeling Process. Eräs mallipohjaisen systeemisuunnittelun menetelmä.
TPM	Technical Performance Measures. Systeemisuunnittelun apuna käytettävä tekninen mittari.
UML	Unified Modeling Language. Mallipohjaiseen ohjelmistosuunnitteluun tarkoitettu graafinen mallinnuskieli.
XMI	XML metadata interchange. Metadatan siirtoprotokolla.
XML	Extensible Markup Language. Tiedonvälitykseen käytettävä kieli järjestelmien välillä.

1. JOHDANTO

Siihen kuinka paljon ihminen on halukas käyttämään aikaansa oppimiseen, on motivaatiolla suuri merkitys [1]. Laboratoriotyöt ja demoharjoitukset kuuluvat olennaisena osana opiskeluun niin opiskelijan motivoimisessa kuin opetustavoitteiden saavuttamisessa. Ne antavat usein myös käsityksen siitä kuinka teoriassa opetettavia asioita sovelletaan käytännössä. Näiden asioiden takia laboratorioharjoituksiin käytettävän oppimisympäristön kehittäminen on tärkeä asia.

Opetushallituksen mukaan oppimisympäristö on fyysisen ympäristön, psyykkisten tekijöiden ja sosiaalisten suhteiden kokonaisuus [2]. Perinteinen oppimisympäristö yliopistomaailmassa on luentosali, jossa suuri joukko opiskelijoita suorittaa oppimissuoritteen kuuntelemalla luennoitsijan opetusta. Tämä on tyypillinen opettajakeskeinen oppimisympäristö. Opiskelijakeskeisessä oppimisympäristössä opiskelija on itse aktiivisemmassa roolissa oppimisen suhteen ja opettaja on enemmän opiskelua tukevassa roolissa. [3]

Realistinen opiskelijakeskeinen oppimisympäristö toimii erinomaisena opettamisen ja oppimisen välineenä opiskeltaessa teollisia prosesseja ja automaatiotekniikkaa. Tällaisessa oppimisympäristössä opiskelija voi aktiivisesti havainnoida prosessien toimintaa itsenäisesti, pienessä ryhmässä tai ohjaajan avustuksella.

Systeemitekniikan laitoksen opetus- ja tutkimuskäyttöön tarkoitettu kolonni on otettu käyttöön vuonna 1983 siihen liittyvän diplomityön myötä [4]. Vuosien varrella tislauskolonnilaboratoriota on kehitetty teknisesti muun muassa opinnäytetöissä [5,6,7].

Tislausprosessin ympärille rakentuvassa oppimisympäristössä on suoritettu vuosien saatossa erilaisia laboratorioharjoituksia prosessien hallintaan ja automaatiotekniikkaan liittyen.Systeemitekniikan laitoksen järjestämän automaation peruskurssiin liittyvä demoharjoitus on järjestetty noin kymmenen opiskelijan ryhmissä. Harjoituksessa opiskelijat ovat tutustuneet laitoksen tislauskolonniin liittyvään prosessiin, sen instrumentointiin ja automaatiojärjestelmään assistentin avustuksella.

Kymmenen henkilön ryhmä on suhteellisen suuri laboratorioharjoitukseen ja ongelmana demoharjoituksessa onkin ollut henkilökohtaisen oppimissuorituksen puuttuminen. Ongelmina suurissa ryhmissä voi esimerkiksi olla vaikeus osallistua aktiivisesti toimintaan, mielipiteiden esittämisen arkuus tai se, että joukkoon on helppo piiloutua ja puuhata omiaan [3]. Jos nykyisen oppimisympäristön puitteissa demoharjoitus järjestetään pie-

nemmissä, esimerkiksi neljän opiskelijan ryhmissä, se tarkoittaisi harjoituksen valvomiseen ja opastukseen liittyvän työn yli viisinkertaistumista.

On tärkeää päivittää oppimisympäristö ja löytää järkevä menetelmä automaation peruskurssin demoharjoituksen suorittamiseksi niin, että se on sujuvaa ja onnistuu ilman assistentin läsnäoloa. Lisäksi oppimisympäristön suunnittelussa täytyy ottaa huomioon muut tislauskolonniprosessia käyttävät kurssit ja tutkimustoiminta.

Kolonniin liittyvä automaatiojärjestelmä ei ole enää täysin ajanmukainen nykyisten automaatioratkaisujen kanssa. Lisäksi tislauskolonnin instrumentointi on osittain vanhentunutta ja rikkinäistä. Nämä puutteet täytyy kartoittaa, jonka perusteella voi suunnitella ja tehdä tarvittavat hankinnat monikäyttöistä oppimisympäristöä varten.

Suomessa muita opetuskäyttöön tarkoitteluja tislauskolonneja löytyy ainakin ammatillisesta WinNova oppimiskeskuksesta sekä Keski-Uudenmaan ammattiopistolta Keudasta [8,9]. Keudan kemian osastolla prosessinhoitajiksi opiskelevat voivat tutustua tislausprosessiin opetuskäyttöön tarkoitetun kolonnin ja siihen liittyvän graafisen käyttöliittymän avulla [9].

Oppimisympäristöjä on tutkittu ja kehitetty opinnäytetöissä jonkin verran. Rask on tutkinut diplomityössään oppimisympäristöjä tietoverkoissa ja suunnitellut oppimisympäristön panosautomaation verkkokurssille [10]. Tampereen teknillisen yliopiston Ohjelmistotekniikan laitoksen kurssien harjoitukset tarvitsivat verkkopohjaista oppimisympäristöä, jota varten kehitettiin IDLE-oppimisympäristö [11]. Teollisiin prosesseihin liittyvää fyysisistä oppimisympäristöä on kehitetty Honkasen opinnäytetyössä. Siinä on suunniteltu ja kehitetty vesiprosessiin liittyvää oppimisympäristöä [12].

Muita Suomessa sijaitsevia prosessiteollisuuteen ja automaatiotekniikkaan liittyviä oppimisympäristöjä ovat muun muassa Chemplant ja Akkulaboratorio, jotka ovat Kokkolan yliopistokeskus Chydeniuksen ja Centria ammattikorkeakoulun yhteisiä tutkimuslaboratorioita. Niitä käytetään kemiantekniikan opiskelussa ja tutkimuksessa, sekä prosessi- ja automaatiotekniikan koulutuksessa. [13]

Tekninen ja tietotekninen kehitys on mahdollistanut oppimisympäristöjen valtavan kehittymisen. Esimerkiksi tietoverkkojen hyväksikäyttö oppimisympäristöjen yhteydessä on lisääntynyt jokaisella opintojen asteella. Varsinkin korkeakoulutasoinen opetus on ottanut tietoverkkoihin perustuvat oppimisympäristöt laajaan käyttöön. Lähes jokaisen yliopistokurssin yhteydessä ainakin osa tiedonvaihdosta käydään verkon välityksellä.

Tietoverkkojen myötä suosioita ovat saavuttaneet myös massiiviset avoimet verkkokurssit (MOOC, massive open online course). Yksi suosituimmista MOOC-kursseja tarjoavista palveluista coursera.org tarjoaa poikkitieteellisesti suuren määrän avoimia

verkkokursseja monista maailman suosituimmista ja arvostetuimmista oppilaitoksista. Tällä hetkellä palvelulla on lähes 7 miljoonaa käyttäjää ja se tarjoaa yli kuusisataa erilaista kurssia yli sadan yhteistyökumppanin toimesta [14]. Lisäksi palvelu tarjoaa mahdollisuuden hankkia verifioitu sertifikaatti kurssin suorittamisesta maksua vastaan. Näin kurssin voi sisällyttää esimerkiksi opintoihin jossain oikeassa yliopistossa tai osoittaa pätevyytensä aihealueesta mahdollisessa työnhakutilanteessa.

MOOC-kurssit mahdollistavat ison opiskelijamäärän sisältävän kurssin tai harjoituksen läpiviennin. Tietynlaisiin aihealueisiin massiiviset verkkokurssit todennäköisesti toimivat, mutta jos tarpeena on tutustuttaa opiskelija käytännön kautta aihealueeseen, pelkkä online-vaikuttaminen on huono vaihtoehto. Lisäksi sosiaalinen kanssakäyminen jää vähäiselle tasolle.

Automaation peruskurssin laboratorioharjoituksen tarkoituksena on tutustuttaa opiskelijat oikean teollisen prosessin ympäristöön sekä siihen liittyvään automaatiotekniikkaan. Tehtävinä on pienessä ryhmässä toimien esimerkiksi tunnistaa PI-kaavion avulla prosessiin liittyviä osia, toimilaitteita ja antureita sekä tutustua prosessin automaatiojärjestelmään. Näiden oppiminen esimerkiksi pelkästään kuvien perusteella olisi hankalaa. Pienessä ryhmässä toimiminen tuo mukanaan sosiaalisen näkökulman, mutta jokaiselle ryhmän jäsenelle jää silti sopivasti tilaa henkilökohtaiseen oppimiseen. Niemistön mukaan 3-5 henkilön ryhmissä vuorovaikutus on helpompaa ja yhteinen toiminta kehittyy nopeammin [15].

Kuusikorven mukaan fyysisen oppimisympäristön suunnittelu ja kehittäminen on parhaimmillaan monimutkaisen, moniammatillisen ja verkostomaisen yhteistyön tulos [16]. Myös simuloinnin hyväksikäyttö voi toimia hänen mukaansa mainiona apuvälineenä oppimisympäristöjä suunniteltaessa [16].

Tässä diplomityössä toteutetaan tislauskolonniin liittyvän oppimisympäristön kehittämisprojekti systeemisuunnittelun menetelmin. Systeemisuunnittelun lähtökohta on toimia monimutkaisten systeemien suunnittelutyössä [17]. Nykyaikaiset mallipohjaisen systeemisuunnittelun (MBSE) työkalut mahdollistavat myös suunniteltavan systeemin mallin rakentamisen ja sen simuloinnin[18,19,20].

Päätarkoituksena on luoda oppimisympäristö, jossa opiskelijat voivat suorittaa automaation peruskurssiin liittyvän laboratorioharjoituksen omatoimisesti. Tämän lisäksi oppimisympäristön kehittämisessä otetaan huomioon myös muut kolonniympäristöä hyväksikäyttävät kurssit sekä mahdollinen tutkimustyö. Näiden vaatimusten perusteella määritetään, mitä asioita ja laitteita kolonniympäristössä täytyy päivittää, hankkia tai muuttaa, että voidaan saavuttaa yleiskäyttöinen teollisten prosessien ja automaatiotekniikan oppimisympäristö.

Systeemisuunnittelun menetelmät ottavat suunnitteluvaiheen aikana huomioon systeemin eri sidosryhmät ja systeemin elinkaaren. Näiden menetelmien avulla voidaan suunnitella ja toteuttaa yleiskäyttöinen oppimisympäristö teollisten prosessien ja automaatiotekniikan opiskeluun. Systeemin elinkaaren huomioon ottaminen ja suunnittelun aikainen dokumentointi ovat tärkeässä asemassa systeemisuunnittelussa. Näiden asioiden pohjalta myös oppimisympäristön ja siihen liittyvän laitteiston ylläpito, huolto ja päivittäminen yksinkertaistuvat.

Tämän työn luvussa 2 käsitellään systeemisuunnittelua ja sen pääperiaatteita. Luvussa 3 perehdytään mallipohjaiseen systeemisuunnitteluun, sen yleisimpiin menetelmiin ja SysML-mallinnuskieleen. Luvussa 4 tutustutaan automaatiojärjestelmiin. Luvussa 5 esitellään tutkimuksen lähtökohdat. Luku 6 määrittää kuinka päädyttiin työssä käytettäviin menetelmiin, miten niitä käytettiin työn edetessä ja kuinka tislauskolonnilaboratoriota muutettiin. Viimeisessä luvussa 7 arvioidaan, kuinka kyseiset tutkimusmenetelmät sopivat tämänkaltaiseen projektiin.

2. SYSTEEMISUUNNITTELU

Sanalla systeemi voidaan tarkoittaa useita eri asioita. Systeemit voivat olla luonnollisia tai ihmisten valmistamia. Usein puhutaan myös termistä järjestelmä. On olemassa esimerkiksi sosiaalisia systeemejä, ekosysteemejä sekä erilaisia fysikaalisia systeemejä. Erilaisten systeemien analysointiin keskittynyttä tieteenalaa kutsutaan systeemiteoriaksi.

Kansainvälisessä standardissa ISO/IEC 15288 määritellään, että systeemi tarkoittaa ihmisten tekemää, luomaa ja käyttämää systeemiä, joka tuottaa palveluita tai tuotteita ennalta määrätyssä ympäristössä sidosryhmien ja käyttäjien hyväksi. Systeemeihin liittyy yksi tai useampia seuraavista elementeistä: laitteita, ohjelmistoja, tietoa, ihmisiä, prosesseja, ohjeita, tiloja, menettelytapoja, materiaaleja. Käytännössä puhutaan usein tuotteesta tai palvelusta, kun tarkoitetaan tämän kaltaista systeemiä. Tässä työssä tarkoitetaan standardin määrittelemää kokonaisuutta systeemistä puhuttaessa. [37]

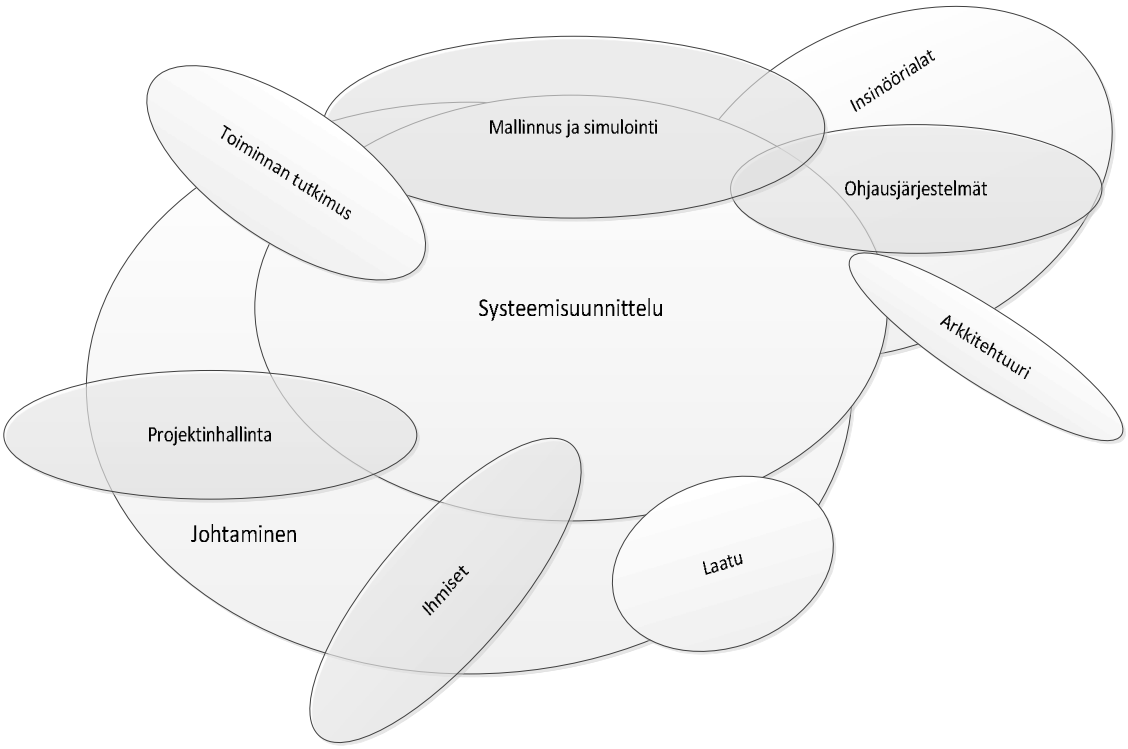
Tässä luvussa käydään aluksi läpi systeemisuunnittelun tarkoitusta ja selvitetään miksi sitä tarvitaan. Seuraavaksi perehdytään systeemisuunnittelun historiaan ja seikkoihin, jotka ovat johtaneet systeemisuunnittelun kehittymiseen. Lopuksi tutustutaan systeeminsuunnittelun elinkaariprosesseihin sekä yleisimpiin työvaiheisiin systeemisuunnitteluprojektissa.

2.1 Systeemisuunnittelun tarkoitus

Nykypäivänä monet insinöörityötä vaativat projektit tapahtuvat poikkitieteellisessä ympäristössä. Useisiin projekteihin liittyy osaajia eri aloilta. Osaamisalueet voivat poiketa toisistaan paljonkin riippuen suunniteltavan systeemin käyttötarkoituksesta ja luonteesta. Projektien laajuus, poikkitieteellisyys, niiden elinkaaren ymmärtäminen sekä vaihtelevat toimintaympäristöt tekevät projekteista erittäin kompleksisia. Useat suunnitteluprojektit vaativat yleensä monen sidosryhmän huomioonottamisen. Systeemisuunnittelu (*Systems Engineering*) on systeemiajatteluun perustuva työtapa, jonka avulla voidaan hallita edellä mainittujen kaltaisia monimutkaisia projekteja [21]. Systeemisuunnittelun prosesseja noudattamalla pystytään tuottamaan järjestelmiä, jotka täyttävät niille määritetyt vaatimukset.

Kilpailulliset syyt vaikuttavat siihen, että tuotteilta vaaditaan entistä halvempaa hintaa ja lyhyempää toimitusaikaa. Myös systeemien väliset riippuvuussuhteet vaikuttavat siten, ettei yhtä systeemiä voi enää ajatella yksinäisenä kokonaisuutena vaan sen suunnittelussa täytyy ottaa huomioon myös siihen liittyvät ulkopuoliset systeemit sekä toimintaympäristö. Näihin asioihin liittyvät ratkaisut sisältävät usein laitteistoja, ohjelmistoja, tie-

toa, ihmisiä sekä palveluja [17]. Säännönmukaiset työtavat suunnitteluprojektissa vähentävät myös mahdollisia suunnitteluvirheitä, jotka kasvattavat onnettomuusriskejä [26].



Kuva 1. *Systeemisuunnittelun poikkitieteellisyys [22]*

Systeemisuunnitteluinsinöörin täytyy hallita systeemisuunnitteluosaamisen lisäksi ainakin pohjatiedot usealta alalta. Tätä vaaditaan siksi, että pystytään havaitsemaan ja hallitsemaan eri alisysteemien sekä komponenttien välisiä rajapintoja.

Systeemisuunnittelu ottaa koko insinööriprojektiin kokonaisvaltaisen näkökulman. Suunnitteluvaiheessa olevaa systeemiä tarkastellaan sisä- ja ulkopuolelta sekä otetaan huomioon systeemin toimintaan liittyvät ulkopuoliset systeemit sekä systeemin toimintaympäristö. Systeemin suunnittelussa on mukana usein eri insinöörialojen osaajia ja asiantuntijoita. [25]

Edellä mainitut seikat muodostavat suunnitteluprojekteista usein hyvin kompleksisia. Systeemisuunnittelun prosessit pyrkivät siihen, että tätä kompleksisuutta pystytään hallitsemaan ja tuottamaan systeemejä nopeasti sekä kustannustehokkaasti. Systeemisuunnittelija määrittää asiakkaan vaatimusten perusteella systeemille vaatimukset ja dokumentoi ne. Vaatimusten pohjalta voidaan tehdä suunnitelma systeemin toteutuksesta. Suunnitelman perusteella toteutetaan osakomponenttien valmistus. Lopuksi valvotaan, että alkuperäiset vaatimukset täyttyvät systeemin ja sen osakomponenttien kohdalla.

Koko suunnitteluprojekti toteutetaan ottamalla huomioon systeemin elinkaaren eri vaiheet. [23]

2.2 Systeemisuunnittelun historia

Teollinen vallankumous toi mukanaan ihmisten valmistamia systeemejä, joiden voidaan katsoa kuuluneen systeemisuunnittelua sisältäviksi projekteiksi. Esimerkkeinä ovat erilaiset työkoneet, autot ja lentokoneet. Kuitenkin tuolloin johtava insinööri pystyi hallitsemaan koko kehitysprojektia, koska projektien osa-alueet pystyttiin tekemään itsenäisesti alueiden välisistä selvistä rajapinnoista johtuen. [23]

Terminä systeemisuunnittelu voidaan katsoa olevan lähtöisin 1940-luvulta, jolloin sitä sovellettiin Bellin laboratorioissa. Bell oli luultavasti ensimmäinen taho, joka alkoi käyttää nimitystä systeemisuunnittelu. Heidän haasteensa kansallisen puhelinverkon rakentamisessa olivat hyvin kompleksisia ennen kuin samankaltaisia tapauksia alkoi esiintyä muiden teollisuudenalojen parista. [24]

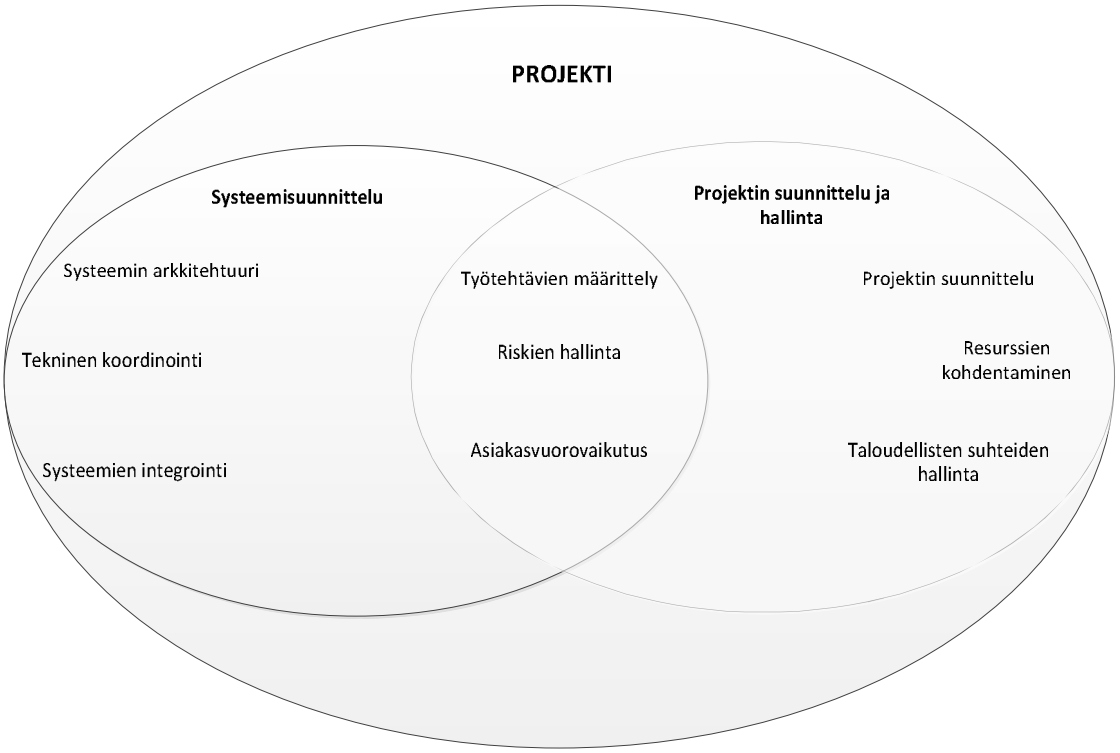
Suurvaltojen varustelukilpailu toisen maailmansodan jälkeen toi mukanaan useita vaativia ja monimutkaisia suunnittelutehtäviä, joiden myötä systeemisuunnittelun prosesseja ja standardeja alettiin kehittää. Lentokonetekniikassa, avaruustekniikassa ja sotateollisuudessa oli tarve kehittää täysin uusia ja erittäin kompleksisia systeemejä. Näistä hyvänä esimerkkinä toimii mannertenvälisten ohjusten suunnitteleminen ja valmistus. Projektien nopea valmistuminen ja onnistuminen oli elintärkeää. Tämän myötä nykyiset systeemisuunnittelumenetelmät ja työprosessit alkoivat kehittyä. [23]

Kansainvälinen systeemisuunnittelujärjestö INCOSE perustettiin 1995 Yhdysvaltain kansallisen NCOSE järjestön pohjalta, joka oli perustettu vuonna 1990. Tämän voittoa tavoittelemattoman järjestön tavoitteena on jakaa ja kehittää parhaita systeemisuunnittelun periaatteita toimivien systeemien kehittämiseen. [39]

2.3 Systeemisuunnittelu osana projektinhallintaa

Systeemisuunnittelussa on teknisten asioiden suunnittelun lisäksi mukana johtamista, vaikka joskus ajatellaan systeemisuunnittelun liittyvän vain teknisiin seikkoihin [27]. Systeemisuunnittelun voidaan katsoa olevan monin tavoin projektinhallinnan kaltainen prosessi. Selvin ero projektinhallinnan ja systeemisuunnittelun välillä on, että systeemisuunnittelussa määritetään yleensä aina vaatimukset ja luodaan arkkitehtuuri. Projektinhallinnalla on laajempi, mutta olennaisesti vähemmän luova rooli. Stevensin, Brookin, Jacksonin ja Arnoldin mukaan projektinhallinta ilman systeemisuunnittelua on merki-

tyksetöntä. Projektinhallinnan ja systeemisuunnittelijoiden yhteistyö täytyy olla saumatonta ja jatkuvaa. [27] Kuvassa 2 on hahmoteltu projektinhallinnan ja systeemisuunnittelun välistä yhteyttä.



Kuva 2. *Systeemisuunnittelun ja projektinhallinnan yhteys [25].*

Systeemisuunnittelu ei myöskään ole yksittäisen erikoisosaajan rooli, vaan se on osa jokaisen suunnitteluprojektiin liittyvän yksilön työtä. Pienissä projekteissa projektipäällikkö voi tehdä systeemisuunnittelun ja valvoa implementoinnin. Isommissa, laajemmissa ja monimutkaisemmissa projekteissa systeemisuunnittelu tapahtuu monella tasolla ja läpi koko kehitysprosessin. [27 s.7]

2.4 Systeemisuunnittelun standardit

Systeemisuunnitteluun on käytössä kolme määriteltyä standardia. [28]

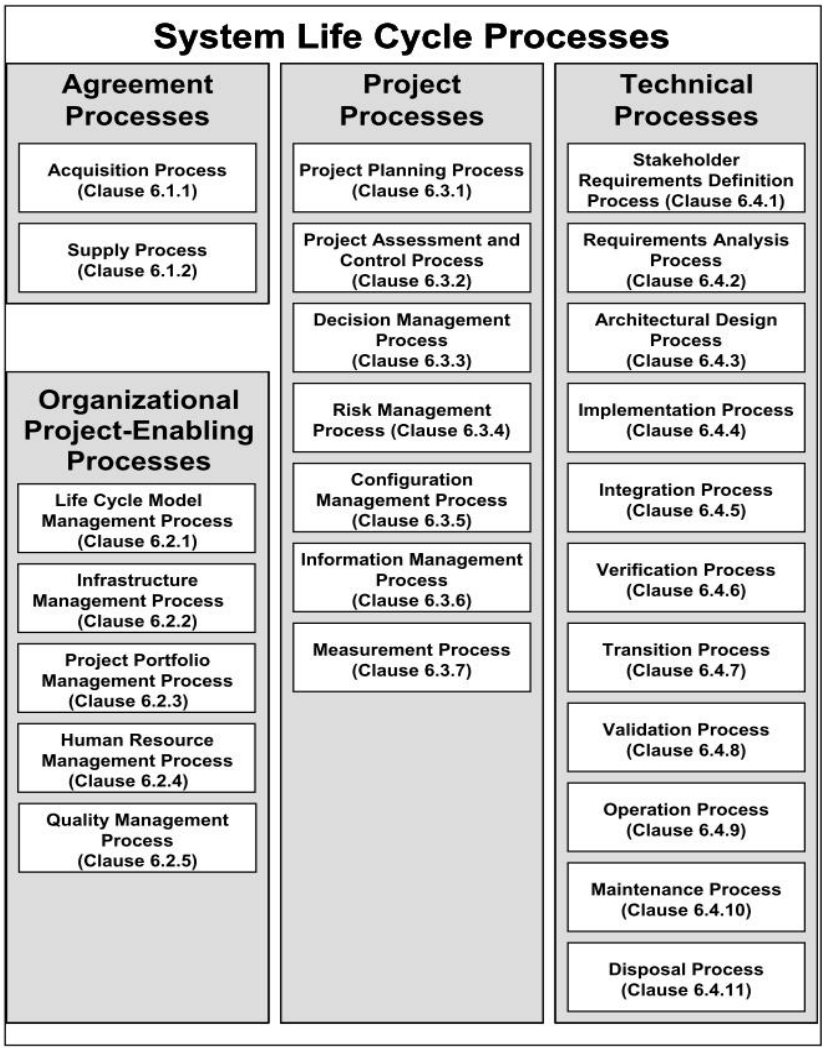
- EIA 632 Processes for engineering a system
- IEEE 1220 Standard for Application and Management of the Systems Engineering Process
- ISO/IEC 15288:2008 International Standard, Systems and software engineering – System life cycle processes

Systeemisuunnittelustandardi ISO/IEC 15288 käsittelee systeemisuunnittelun elinkaariprosesseja. ISO/IEC 15288 on standardeista kattavin sisältäen systeemin koko elinkaa-

ren prosessit. Kyseinen standardi jakaa systeemisuunnitteluprosessit neljään pääkategoriaan:

- sopimusprosessit (Agreement process)
- organisaation tukiprosessit (Organizational Project-Enabling Processes)
- projektiprosessit (Project process)
- tekniset prosessit (Technical process)

Nämä pääprosessit koostuvat eri osaprosesseista, joita on yhteensä 25 kappaletta. Useimmat osaprosessit kuuluvat teknisiin prosesseihin. [28] ISO/IEC 15288 sisältämät osaprosessit ovat lueteltuna kuvassa 3. Prosessilla tarkoitetaan toimintojen kokonaisuutta, joiden avulla saavutetaan ennalta määrätyistä sisääntuloista haluttuja ulostuloja. [37]



Kuva 3. *Systeemin elinkaariprosessit [37].*

2.5 Systeemin elinkaari

Jokaisella ihmisen valmistamalla systeemillä on elinkaari, vaikka sitä ei aina olisi määriteltykään. Jokaisen systeemin tai tuotteen elinkaari koostuu liiketoimintanäkökulmasta, rahoitusnäkökulmasta ja teknisestä näkökulmasta. Elinkaaren eri vaiheissa täytyy ottaa huomioon nämä kaikki näkökulmat tehdessä päätöstä elinkaaren seuraavaan vaiheeseen siirtymisestä. Systeemisuunnittelija luo teknisen ratkaisun, joka täyttää liiketoiminnan ja rahoituksen reunaehdot. [21]

Systeemin elinkaaren ymmärtäminen on yksi tärkeä osa systeemisuunnittelua. Systeeminsuunnittelu prosessit ottavat huomioon elinkaaren systeemin kehityksessä tarpeen tasolta aina sen mahdolliseen hävittämiseen asti. Elinkaari täytyy ottaa huomioon päätöksenteossa koko suunnitteluprojektin ajan. Systeemin elinkaari alkaa systeemin määrittelystä ja sen suunnittelusta. Elinkaari jatkuu rakentamisen ja tuotannon läpi toimintaan, kunnossapidosta ja ylläpidosta aina systeemin toiminnan lopettamiseen tai hävittämiseen asti [17].

2.5.1 Systeemin elinkaaren vaiheet

Systeemin elinkaaren vaiheet on määritelty ISO/IEC 15288 standardissa kuudessa osassa. Nämä vaiheet ovat kuvattu taulukossa 1. Eri vaiheissa toteutetaan kuvassa 3 lueteltuja systeemisuunnittelun prosesseja. INCOSE kuitenkin korostaa, että jokaisen organisaation täytyy itse valita prosessit ja toiminnot kullekin elinkaaren vaiheille, jotka parhaiten sopivat kuhunkin projektiin. Jokaisen vaiheen välillä suoritetaan päätöksenteko voidaanko edetä elinkaaren seuraavaan vaiheeseen. Myös koko projektin lopettaminen on mahdollista. [21]

Taulukko 1. *ISO/IEC 15288 mukaiset systeemin elinkaaren vaiheet[21].*

Elinkaaren vaiheet	Tarkoitus	Päätösportit
Konsepti	Tunnista sidosryhmien tarpeet Tutki konsepteja Tee toteuttamiskelpoisia vaihtoehtoja	Päätösvaihtoehdot: -siirry seuraavaan vaiheeseen -jatka saman vaiheen suorittamista -palaa edelliseen vaiheeseen -keskeytä projekti -lopeta projekti
Kehitys	Määritä systeemivaatimukset Luo kuvaus toteutuksesta Integroi systeemi Verifioi ja validoi systeemi	
Tuotanto	Valmista systeemi Seuraa ja testaa	
Käyttö	Operoi systeemiä asiakkaan vaatimusten perusteella	
Ylläpito	Pidä yllä systeemin toimintakykyä	
Käytöstä poisto	Varastoi, arkistoi tai hävitä systeemi	

Konseptivaiheessa määritetään sidosryhmien vaatimukset, tutkitaan systeemin konseptia ja esitetään toteutuskelpoisia ratkaisuja. Tässä vaiheessa voidaan luoda malleja ja prototyyppejä sekä tehdä simulointeja erilaisten ratkaisujen vertailemisen avuksi. Konseptivaiheessa luodaan määritelmä systeemin, alisysteemien ja arkkitehtuurin konseptista. Samalla luodaan edellä mainittujen osien integraatio-, verifikaatio- ja validointisuunnitelmat. [21]

Kehitysvaiheessa luodaan lopulliset systeemivaatimukset, laaditaan ratkaisun dokumentointi sekä toteutetaan alisysteemit ja osakomponentit. Lopuksi suunniteltu systeemi verifioidaan ja validoidaan. Verifioinnissa tutkitaan onko toteutus sellainen, mitä on suunniteltu. Validointi tarkoittaa sitä, että tutkitaan sopiiko toteutus kyseiseen asiaan.

Tuotantovaiheessa suunnittelun kohteena oleva systeemi valmistetaan sille suunniteltuun toimintaympäristöön. Tässä vaiheessa voidaan joutua tekemään muutoksia. Syyt muutoksiin voivat olla esimerkiksi: ongelmat valmistuksessa, tuotantokustannukset tai tuotettavan systeemin suorituskyvyn parantaminen. Muutokset voivat muuttaa systeemivaatimuksia. Myös verifiointi ja validointi voidaan joutua tekemään uudelleen. [21]

Käyttövaiheessa suunniteltu systeemi toimii ennalta määritetyssä toimintaympäristössä ja se tuottaa haluttuja palveluita. Tässäkin vaiheessa voidaan tehdä muutoksia esimerkiksi systeemin ominaisuuksien parantamiseksi. Muutoksien teossa täytyy noudattaa

systeemisuunnittelun prosesseja, että voidaan varmistaa muutoksien sopivuus systeemin toimintaan. [21]

Ylläpitovaiheessa systeemiä pidetään toimintakykyisenä. Systeemiin voidaan joutua tekemään muutoksia esimerkiksi käyttökustannusten pienentämiseksi tai systeemin käyttöiän pidentämiseksi. Mahdolliset muutokset vaativat systeemisuunnittelun ottamista huomioon, ettei systeemi menetä alkuperäisiä ominaisuuksiaan. [21]

Alasajovaiheessa systeemin toiminta lopetetaan ja se varastoidaan tai tuhotaan. Systeemisuunnittelun toiminta, joka tässä vaiheessa suoritetaan, on varmistamista, että alasajovaiheeseen liittyvät vaatimukset tulevat täytettyä. Vaatimukset jotka liittyvät systeemin alasajoon, määritetään ja selvitetään jo systeemin konseptivaiheessa. [21]

Elinkaaren eri vaiheissa voi olla päällekkäisyyttä ja samanaikaisuutta. Esimerkiksi käyttö- ja ylläpitovaihe ovat käytännössä aina samanaikaisia. ISO/IEC 15288 standardin eri prosessit ja toiminnot pyrkivät siihen, että eri elinkaarivaiheiden päämäärät voidaan saavuttaa [21].

Elinkaaren ymmärtäminen ja jokaisen vaiheen huomioon ottaminen on tärkeää. Yhden elinkaarivaiheen ohittaminen systeemin suunnittelussa voi tulla erittäin kalliiksi myöhemmässä vaiheessa. Elinkaaren suunnitteluun on olemassa erilaisia suunnittelumalleja. Mallit keskittyvät usein elinkaaren konsepti- ja kehitysvaiheeseen ja sisältävät teknisiä prosesseja. [21]

2.6 Erilaiset suunnittelumallit

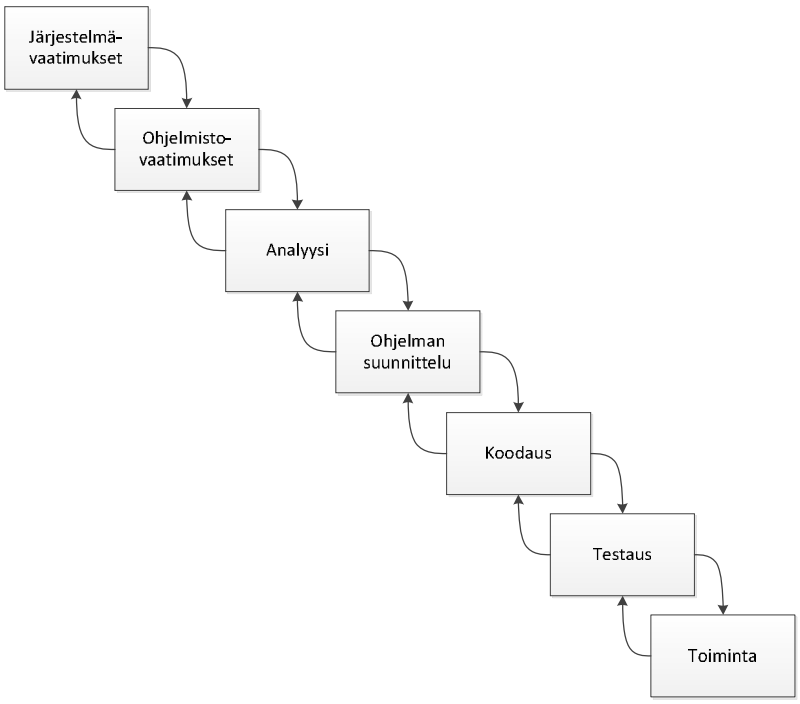
Monet systeemisuunnittelun mallit ovat lähtöisin ohjelmistosuunnittelun parista. Mallit ovat usein graafisia kuvauksia siitä kuinka eri prosessit suunnitteluprojektissa suoritetaan systeemin elinkaaren vaiheissa. Suunnittelumallit kuvaavat yleensä systeemisuunnittelun työvaiheita pelkästään systeemin teknisen kehityksen aikana. Suunnittelumalleissa kuvatut systeemisuunnittelun työvaiheet sisältyvät suurilta osin ISO/IEC 15288 standardin teknisiin osaprosesseihin sekä elinkaaren vaiheista konsepti- ja kehitysvaiheeseen.

Ensimmäiset suunnittelumallit olivat hyvin lineaarisia suunnittelun etenemisen kannalta. Paljon käytettyjä suunnittelumalleja ovat esimerkiksi vesiputousmalli, spiraalimalli ja V-malli. Nykyisiin suunnittelumalleihin liittyy vahvasti iteratiivisuus eri vaiheiden välillä.

2.6.1 Vesiputousmalli

Vesiputousmalli on alun perin tarkoitettu ohjelmistotuotantoon. Mallin pääajatuksen esitteli Royce artikkelissaan *Managing the Development of Large Software Systems* jo

vuonna 1970 [29]. Malli nimettiin vasta myöhemmin vesiputousmalliksi. Vesiputousmalli esittää ohjelmistoprojektin vaiheet peräkkäisinä elinkaaren vaiheina. Vesiputousmallia käytettiin suunnittelumallina myös teollisuusprojekteissa. Nämä projektit yksinkertaistuivat usein kuitenkin niin, että iterointi eri suunnitteluvaiheiden välillä jäi puuttumaan. [30 s.37]



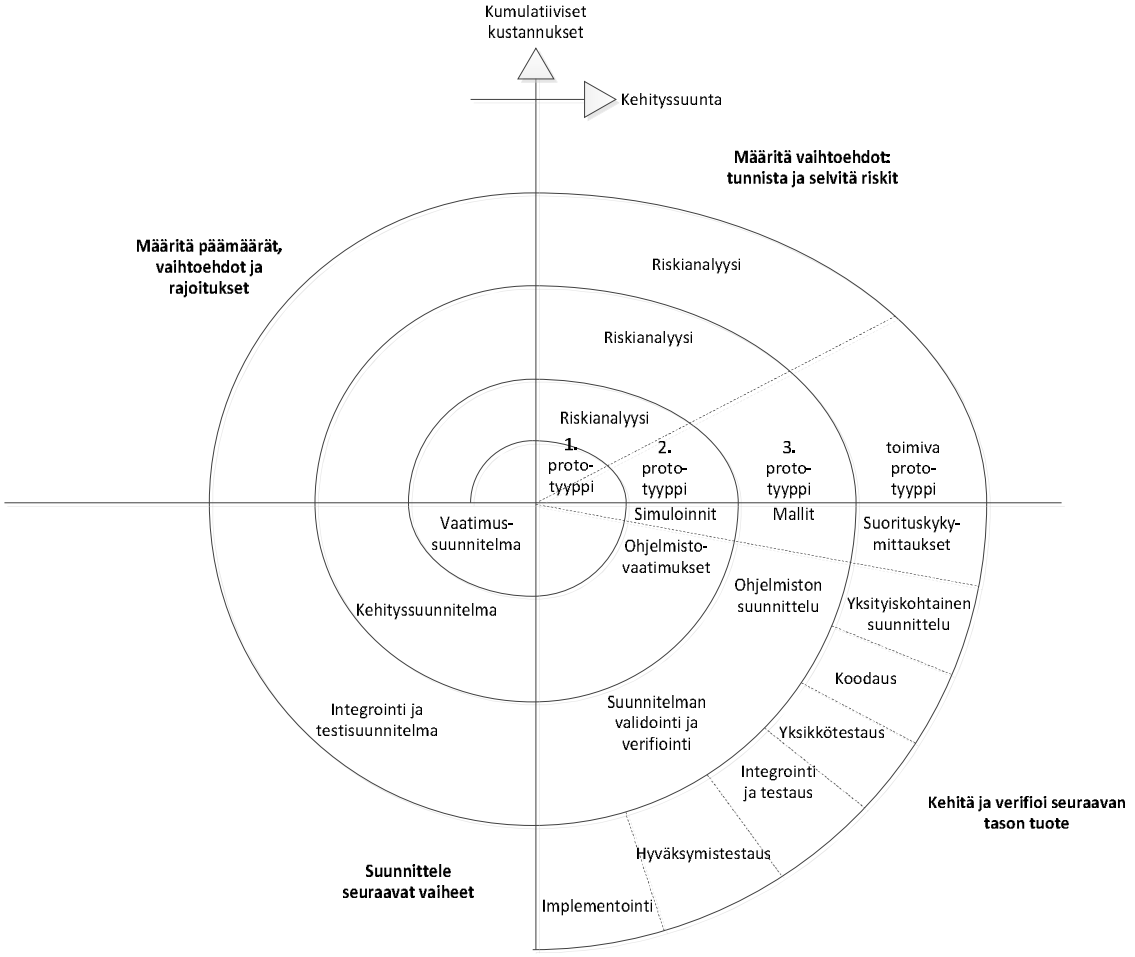
Kuva 4. *Vesiputousmalli [29].*

Vaiheiden määrä ja tarkat kuvaukset voivat mallissa olla erilaisia käyttökohteesta riippuen. Perusidea on kuitenkin sama. Ohjelmistotuotannon vesiputousmallissa lähdetään liikkeelle järjestelmän vaatimuksista. Niiden perusteella tehdään ohjelmistovaatimukset, analysointi ja suunnitellaan ohjelma. Suunnitelman perusteella koodataan ja testataan, jonka jälkeen saavutetaan ohjelma toimintakunnossa. Suurin ongelma vesiputousmallissa on, että iterointi vaiheiden välissä on liian yleisellä tasolla [22]. Royce itse näki tässäkin mallissa heikkoutena sen, että vaatimuksien muokkaaminen oli hankalaa testausvaiheessa huomattujen ongelmien jälkeen. Ongelmien paljastuminen testausvaiheessa voi hänen mielestään kaksinkertaistaa kustannukset. [29]

Vaikka vesiputousmallia pidetään usein liian lineaarisena, sitä käytetään muiden suunnittelumallien osana. Esimerkiksi ohjelmistosuunnittelussa käytetty ketterä Scrum-malli voidaan katsoa pitävän sisällään useita pieniä vesiputousmalleja. [30 s.37]

2.6.2 Spiraalimalli

Spiraalimallin esitteli Boehm vuonna 1986 [31]. Malli on alun perin tarkoitettu ohjelmistosuunnitteluun. Malli on yhdistelmä lineaarisesta kehitysmallista ja iteratiivisesta mallista. Spiraalimalli jakaa systeemin kehityksen neljään perusosaan: tavoitteiden ja rajoitteiden määrittely, vaihtoehtojen ja riskien arviointi, kehittäminen ja seuraavan tason tuotteen määrittely, seuraavien vaiheiden suunnittelu. Spiraalin kierrosten lukumäärä kuvaa iterointikertoja systeemin suunnittelussa. Jokainen kierros lisää luonnollisesti kustannuksia suunnitteluprojektissa, mutta järjestelmän toimintavarmuus kasvaa ja riskit pienenevät. Iterointikierrosten määrä voidaan määrittää systeemi- tai ohjelmistosuunnittelijan tarpeen perusteella. Viimeisellä spiraalin kierroksella integroidaan alkuperäisten vaatimusten perusteella toimiva ja valmis lopputuote. Spiraalimallin graafinen esitys on esitelty kuvassa 5. [22]

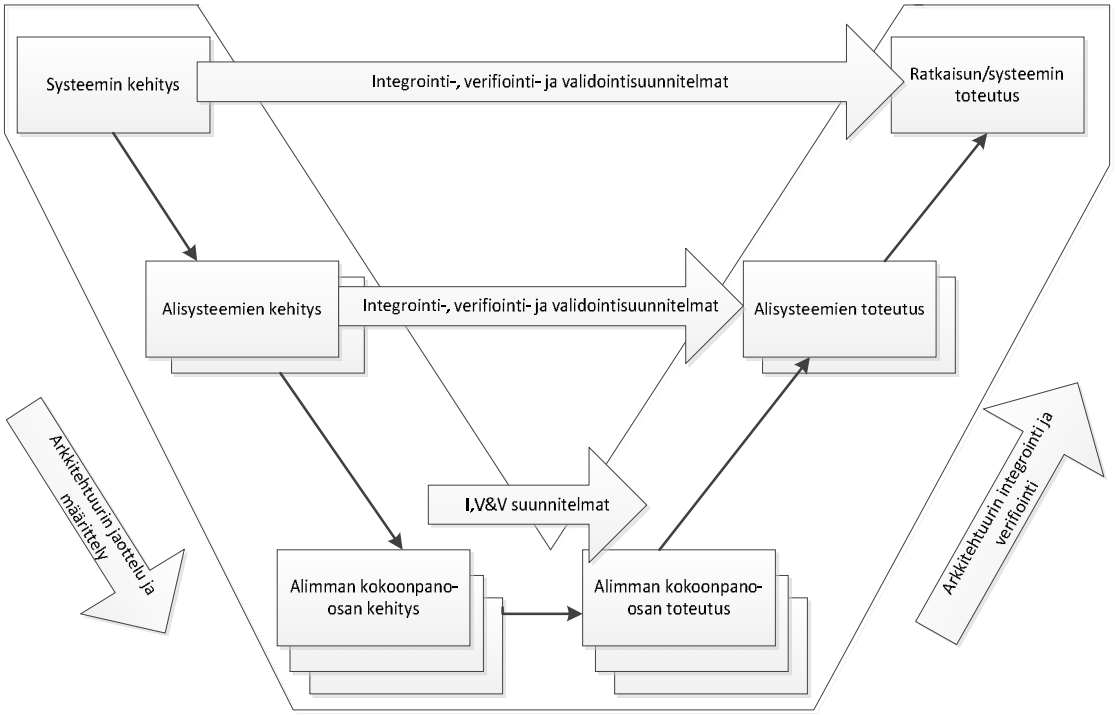


Kuva 5. *Spiraalimalli [22].*

2.6.3 V-malli

V-malli on käytännöllinen kuvaus systeemisuunnittelutapahtumista. ISO/IEC 15288 elinkaaren vaiheista V-malli pitää sisällään konseptivaiheen ja kehitysvaiheen prosesse-

ja. Malli kuvaa projektia sen määrittelystä aina verifiointiin saakka. Kirjallisuudessa V-mallin esitteli ensimmäisenä Rook paperissaan ohjelmistoprojektien kontrolloimisesta vuonna 1986 [32]. Systeemisuunnittelun työkaluna malli esiteltiin Forsbergin ja Moozin paperissa vuonna 1991 [33]. Kuvassa 6 on esitelty yksinkertainen V-malli, jossa suunnitteluvaiheet on esitetty pääpiirteittäin.



Kuva 6. *V-malli, mukailtu lähteestä [34].*

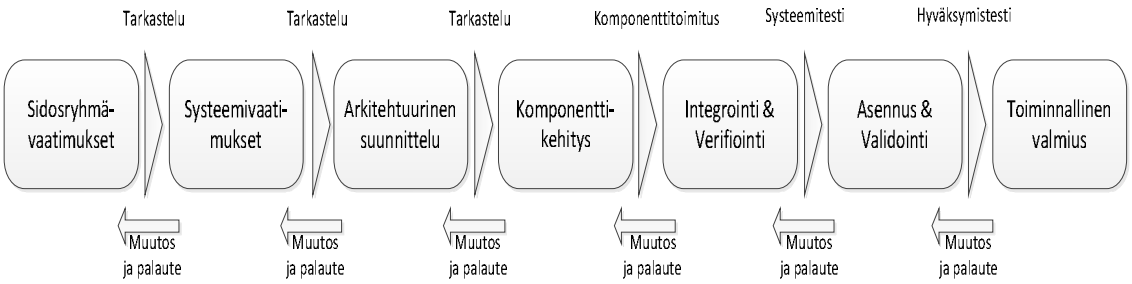
V-mallin vasemmalla sivulla tehdään määritelmät ja suunnitelmat systeemistä. Oikealla sivulla suoritetaan kokoonpano-osien integrointi, testaukset sekä verifiointi ja validointi. V-mallin ensimmäisenä vaiheena on käyttäjien vaatimusten määrittäminen. Käyttäjien vaatimusten perusteella voidaan määritellä systeemin vaatimukset. Systeemivaatimusten perusteella tehdään arkkitehtuurinen suunnittelutyö. Tämän jälkeen kehitetään arkkitehtuuristen suunnitelmien sisältämät komponentit ja osat (configuration items).

Suunnittelutyön jälkeen alisysteemit ja osat toteutetaan. Tämä voi tarkoittaa ohjelmistojen tuottamista tai fyysisten osien valmistusta. Myös valmiiden osien tai osajärjestelmien hankkiminen kuuluu tähän vaiheeseen. Näistä osista suoritetaan systeemin kokoonpano. Lopullisen järjestelmän asennus ja käyttöönoton jälkeen testataan systeemin toimintavalmius. Jokaisen suunnittelutason välillä suoritetaan testaus siitä, kuinka valmiit osat, osajärjestelmät ja lopullinen systeemi vastaa suunniteltua ja täyttyvätkö alkuperäiset vaatimusmäärittelyt.

V-malli on yksi yleisimmin käytettyjä systeemisuunnittelun suunnittelumalleja. Se kuvaa pääpiirteittäin hyvin systeemisuunnittelun konsepti- ja kehitysvaiheen teknisiä prosesseja. Eri vaiheet käytetyissä malleissa voivat erota hiukan toisistaan, mutta tapahtumien kulku on samankaltainen.

2.7 Systeemisuunnitteluprojektin päävaiheet

Riippumatta käytetystä suunnittelumallista systeemisuunnitteluprojektin päävaiheet ovat usein hyvin samankaltaisia. Erilaiset suunnittelumallit ja menetelmät kuvaavat usein ISO/IEC 15288 standardin konsepti ja kehitysvaiheeseen liittyviä prosesseja. Kuvassa 7 on yksinkertainen kuvaus systeemisuunnitteluprojektin päävaiheista.



Kuva 7. *Systeemisuunnitteluprojektin osavaiheet, mukaillen [27].*

Ensin tehdään vaatimusmäärittely, joka yleensä jaotellaan sidosryhmävaatimuksiin ja systeemivaatimuksiin. Systeemivaatimusten perusteella luodaan suunnitelma systeemin arkkitehtuurista, jonka perustella voidaan määrittää alisysteemit ja tarvittavat kokoonpano-osat. Kokoonpano-osat, alisysteemit ja systeemi integroidaan ja verfioidaan. Lopuksi voidaan suorittaa lopullinen asennus toimintaympäristöön ja suorittaa validointi. Tämä luku sisältää yleiset systeemisuunnitteluprojektin päävaiheet sekä niihin liittyvät pääpiirteet.

2.7.1 Vaatimusmäärittely

Systeemisuunnittelun eri standardien suunnittelutavat eroavat joiltain osin toisistaan, mutta yhteisenä piirteenä lähes kaikille on vaatimusten määrittely osana systeemisuunnitteluprosessia. Tätä pidetään usein tärkeimpänä vaiheena suunnitteluprojektissa. Liian kapeasti tehty vaatimusmäärittely voi tulla erittäin kalliiksi myöhemmässä vaiheessa.

Systeemisuunnittelussa vaatimukset jaetaan yleensä sidosryhmävaatimuksiin ja systeemivaatimuksiin. Sidosryhmävaatimuksissa määritetään systeemin tulevien käyttäjien ja sidosryhmien vaatimukset. Näiden vaatimusten perusteella voidaan seuraavassa suunnittelun vaiheessa luoda systeemivaatimukset. [27,35]

Sidosryhmävaatimusten määrittely täytyy tehdä laajasti, mahdollisesti jopa liian laajasti. Puuttuva vaatimus tulee yleensä huomattavasti kalliimmaksi, jos se huomataan vasta systeemin valmistuttua tai suunnitteluprojektin myöhemmissä vaiheissa. Mahdolliset päällekkäisyydet käyttäjävaatimuksissa on helppo havaita jo suunnitteluvaiheessa ja luokitella ne omiin ryhmiinsä. Kaikkien sidosryhmien tunnistaminen on tärkeää. Tunnistamisen jälkeen sidosryhmät on syytä priorisoida ja asettaa tärkeysjärjestykseen. Tätä voidaan käyttää apuna esimerkiksi tehtäessä investointipäätöksiä. [27]

2.7.2 Sidosryhmävaatimukset

Systeemin sidosryhmillä tarkoitetaan ihmisiä, jotka ovat yhteydessä systeemin kehitykseen, käyttämiseen tai joihin systeemin toiminta jollain tavalla vaikuttaa. Sidosryhmät pitävät sisällään esimerkiksi systeemin käyttäjät, ylläpitäjät, rahoittajat, johtajat, kehittäjät ja kriitikot [36 s.38] Mukaan voidaan katsoa lukeutuvan myös tahot, joihin systeemin toiminta vaikuttaa haitallisesti esimerkiksi päästöjen tapauksessa [35]. Sidosryhmät joiden tarpeet liittyvät suunniteltavan systeemin toiminnalliseen käyttöön katsotaan osaksi sidosryhmävaatimuksia. Ne joiden tarpeet liittyvät kehitysympäristöön voidaan määritellä myös systeemivaatimuksiin [27 s.21].

Sidosryhmävaatimuksia tulisi määritellä toiminnalliselta kannalta [27]. Niiden tulisi olla lyhyitä, ei-teknisiä ja siinä muodossa, että systeemin käyttäjien on helppo ne ymmärtää. Vaatimusten ei tule sisältää tietoa siitä kuinka ne tullaan systeemissä lopullisesti toteuttamaan. Vaatimuksia jotka sisältävät suunnitteluideoita Gilb kutsuu ”vääriksi vaatimuksiksi” [36 s.39]. Ne tulisi hänen mukaansa poistaa vaatimusmäärittelystä.

Kriittisten sidosryhmien tunnistamisen epäonnistuminen on yleinen ongelma. Tämä voi aiheuttaa tärkeiden vaatimusten puuttumisen ja vaikuttaa esimerkiksi systeemin kannattavuuteen. Systeemin elinkaaren huomioon ottaminen on tärkeässä roolissa jo sidosryhmien tunnistamisessa. Systeemisuunnittelijan täytyy tunnistaa kaikki systeemiin liittyvät sidosryhmät koko sen elinkaaren aikana. Lisäksi olisi hyvä tunnistaa ja erotella systeemin sisäiset ja ulkoiset sidosryhmät. [36 s.38] Yleensä systeemin loppukäyttäjät ja asiakkaat on helppo tunnistaa, mutta esimerkiksi lainsäädännölliset tahot, jotka voivat vaikuttaa suunniteltavan systeemin toimintaan tulisi tunnistaa [21]. Aina ei ole selvää kenelle kuuluu vastuu esimerkiksi mahdollisessa onnettomuustilanteessa ja siksi kaukaiseltakin tuntuvat sidosryhmät olisi syytä selvittää.

Tärkeiden sidosryhmien lisäksi vaatimustenmäärittelyprosessissa on tärkeää tunnistaa erilaisten vaatimusten prioriteetti. Tärkeimmät vaatimukset ovat niitä, jotka ovat välttämättömiä, tuottavia tai sisältävät suuria riskejä systeemin kannalta. On syytä tarkastella vaatimuksia alueilta, jotka aiheuttavat suurimmat kustannukset kehitykseen tai toimintaan. Tarkastelun jälkeen voidaan keskustella sidosryhmien kanssa mitkä vaatimukset ovat tärkeimpiä ja kriittisimpiä toteuttaa [36 s.39]

Joskus osa käyttäjävaatimuksista voi olla niin kokonaisvaltaisia, että vähemmän tärkeistä joudutaan luopumaan. Tärkeää on myös ajoissa selvittää ovatko vaatimukset edes toteutettavissa. Sidosryhmävaatimusten monimutkaisuustaso on usein matala, eivätkä vaatimukset sisällä tarkkoja toteutukseen liittyviä asioita. Yksinkertaisilta kuulostavat vaatimukset voivatkin olla mahdottomia toteuttaa alisysteemi- tai komponenttitasolla. [34 s.92]

Sidosryhmävaatimusprosessin aikana pitää määrittää myös rajoitukset, joita voi muodostua esimerkiksi lainsäädännön myötä [21]. Systeemi voi sisältää vaarallisia tiloja, joita koskevat omat standardit ja määräykset. Hyviä esimerkkejä ovat räjähdysvaaralliset tilat, joiden suunnittelussa on useita erityispiirteitä.

2.7.3 Sidosryhmävaatimusten lähteet

Sidosryhmävaatimusten hankkimiseen on useita mahdollisia lähteitä ja vaatimusten hankkiminen kannattaa tehdä mahdollisimman laajasti. Stevens et al. [27] luettelee seuraavia lähteitä sidosryhmävaatimuksille:

Haastattelut ovat yleinen sidosryhmävaatimusten lähde. Haastattelut on syytä suunnitella etukäteen ja haastatteluprosessissa täytyy mukana olla henkilöitä, jotka ymmärtävät vaatimusmäärittelyprosessin. Haastattelu ja dokumentointi samaan aikaan voi olla liian työlästä ja tärkeitä yksityiskohtia jää tällöin huomioimatta. Tavoitteena on kirjata kaikki tieto talteen ja suodattaa siitä tarvittava, jotta voidaan muodostaa järkeviä vaatimuksia.

Työskentely käyttäjäympäristössä tuottaa usein hyviä vaatimuksia myös tulevien järjestelmien suunnittelussa. Tämä tuo synergiaetuja kehittäjien ja käyttäjien välille koko kehityskaaren aikana.

Samankaltaisten tai samanlaisten olemassa olevien systeemien esittely oman suunniteltavan systeemin käyttäjille ja heidän haastatteleminen olemassa olevien systeemien hyvistä ja huonoista puolista voi olla hyvä lähde oman systeemin vaatimuksille. Olemassa olevien kehitysvaiheessa olevien samankaltaisten systeemien kehitysehdotukset ja ongelmatapaukset voivat olla myös lähde vaatimuksille. Usein on huomattavasti helpompi löytää vaatimuksia edellisten systeemien ongelmista kuin muodostaa täysin uusia ja käsitteellisiä vaatimuksia.

Käyttäjäryhmien tapaamiset, jotka jo ennestään käyttävät samanlaista systeemiä. Hyvä esimerkki on ohjelmistojen vanhat versiot. Näiden jo kokeneiden käyttäjien ehdotukset ovat yleensä erittäin hyviä vaatimusten lähteitä. Samantyyppisessä tilanteessa myös erilaiset työpajat voivat olla lähde kerätä käyttäjävaatimuksia.

Käyttäjien modifikaatiot jo olemassa oleviin järjestelmiin on yksi merkittävimpiä käyttäjävaatimusten lähteistä. Modifikaatiot voidaan karakterisoida vaatimuksiksi ja muuntaa ne suunniteltavan systeemin määrittelyihin.

Innovaatiot luovat usein puoliksi valmiita tuotteita. Näiden pohjalta voidaan määrittää vaatimuksia, jotka sopivat suunniteltavan tuotteen vaatimuksiin.

Tutkimukset ja erilaiset raportit voivat olla hyvä lähde vaatimuksille. Tällöin vaatimukset täytyy etsiä raporteista ja muuttaa ne enemmän formaaleiksi. Jäljitettävyys vaatimusten ja alkuperäisten materiaalin välillä täytyy kuitenkin säilyttää.

Prototyypit voivat olla mainioita käyttäjävaatimusten lähteitä. Prototyypit voivat olla monimutkaisia, kalliita ja toimivia osatoteutuksia. Yksinkertainen systeemin prototyyppi voi olla esimerkiksi pelkkä simulaatio. Ideana on kuitenkin, että ne voivat auttaa sidosryhmiä havainnollistamaan lopullista systeemiä ja muodostamaan sen perusteella vaatimuksia.

Uusi teknologia voi mahdollistaa aikaisempien vaatimusten toteuttamisen. Vaatimukset jotka olivat aikaisemmin mahdottomia tai kustannuksiltaan liian suuria toteuttaa voivat teknologisen kehityksen myötä tulla toteuttamiskelpoiseksi.

Kyselylomakkeet ovat usein käytettyjä, mutta niiden ei tulisi olla ensisijaisia vaatimuslähteitä. Ne voivat toimia myös hyvänä tapana varmistaa ja tarkentaa vaatimuksia.

Käyttäjävaatimuksia voidaan tarkastella ajattelemalla vaatimusten toimintaa niin, että suunniteltava systeemi olisi valmis. Systeemiä voidaan simuloida käymällä askelittain läpi kuinka se toteuttaa halutut päämäärät luomalla toiminnallisia skenaarioita. Näitä skenaarioita voidaan mallintaa ohjelmistosuunnittelusta tuttujen käyttötapauksien (*Use Case*) avulla.

Käyttötapaukset ovat tapahtumia, jotka voivat toteuttaa usean erilaisen skenaarion. Käyttötapauksia voidaan kuvata pelkästään sanallisella selityksellä tai sitä voidaan lisäksi mallintaa graafisella kaaviolla. Kaavion muodostamiseen voidaan käyttää graafisia mallinnuskieliä, joita ovat esimerkiksi UML ja SysML. Käyttötapaukset pitävät sisällään yleensä yhden päätapahtuman ja erilaisia vaihtoehtoisia toimintakuvauksia. [27]

Vaatimuksen määrittämiseen tarvitaan yleensä useampi skenaario. Skenaarioiden avulla voidaan tunnistaa päällekkäisiä ja samankaltaisia vaatimuksia. Myös mahdolliset puutteet käyttäjävaatimuksissa voidaan havaita ja muokata vaatimuksia kattavimmaksi. Toisensa poissulkevia vaatimuksia ei voida myöskään hyväksyä. Näissä tapauksissa täytyy etsiä sidosryhmien kanssa ratkaisu, joka sopii molemmille tai poistaa toinen. [27]

Vaatimusten keräämisen jälkeen ne ovat todennäköisesti suhteellisen epäorganisoidussa muodossa. Lisäksi monet vaatimukset ovat todennäköisesti usein esitettyjä tai vaatimuksissa on päällekkäisyyttä, vaikka ne eroaisivatkin tarkasti ottaen. Samankaltaiset vaatimukset täytyy tunnistaa ja ryhmitellä omiin kokonaisuuksiinsa. Hyvin ryhmitellyt vaatimukset auttavat myös tunnistamaan mahdollisia päällekkäisyyksiä. Jokaisesta vaatimuksesta täytyy voida vetää yhteys alkuperäiseen käyttäjään, jota vaatimus koskee.

Tätä yhteyttä kutsutaan vaatimusten jäljitettävyydeksi (*traceability*). Kun käyttäjävaatimukset on saatu kerättyä ja analysoitua vaatimukset pitää hyväksyttää sidosryhmillä. Perinteisessä systeemisuunnittelussa tätä vaatimusdokumenttia kutsutaan sidosryhmävaatimusdokumentiksi [17].

2.7.4 Vaatimusten jäljitettävyys

Vaatimusmäärittelyssä jäljitettävyys kuvaa sitä kuinka korkean tason vaatimukset muuntautuvat alempien tasojen vaatimuksiksi. Esimerkiksi systeemivaatimusten täytyy toteuttaa sidosryhmävaatimukset. Systeemivaatimukset jaetaan alisysteemien teknisiksi vaatimuksiksi ja alisysteemeiltä edelleen komponenteille. Systeemin jokaisen komponentin tulisi olla jäljitettävissä aina johonkin sidosryhmävaatimukseen asti. [35]

Systeemivaatimuksia muodostettaessa jokainen käyttäjävaatimus pitäisi olla jäljitettävissä ainakin yhteen systeemivaatimukseen [17 s.41]. Jäljitettävyyttä käyttäjävaatimuksista systeemivaatimuksiin kutsutaan eteenpäin jäljitettävyydeksi. On kuitenkin syytä pitää huolta jäljitettävyydestä myös taaksepäin, joka tarkoittaa jokaisen systeemivaatimuksen jäljitettävyydestä ainakin yhteen sidosryhmävaatimukseen. Taaksepäin jäljitettävyys takaa sen, ettei systeemin konseptin suunnitteluvaiheessa ole tullut ylimääräisiä systeemivaatimuksia, joita sidosryhmät eivät ole hyväksyneet. [17]

Vaatimusten jäljitettävyys tuo mukanaan useita hyötyjä systeemin suunnitteluprojektin edetessä. Kun maalit saavutetaan, on helppoa varmistaa, kuinka ne toteuttavat alkuperäisiä sidosryhmien vaatimuksia siitä miten systeemin tulisi suoriutua. Muutostilanteiden hahmottaminen on helpompaa, kun pystytään selvittämään muutosten vaikutusketju koko systeemin osalta. Kunnollinen jäljitettävyys mahdollistaa selvillä olemisen suunnitteluprojektissa edistymisen tilasta. Myös kustannusten ja hyötyjen välillä tasapainottelu on helpompaa, kun tiedostetaan kuinka komponentit ovat suhteessa vaatimuksiin. [35]

Suunnitellun järjestelmän alihankkijoilla voi olla käsitys, ettei heidän valmistamallaan systeemin osalla ole sidosryhmävaatimuksia. Esimerkiksi auton moottorinohjauspiirien valmistaja voi ajatella, etteivät auton käyttäjän vaatimukset koske moottorinohjauspiiriä. Vaatimusten täytyy kuitenkin kulkea systeemin, sen alisysteemien ja osakomponenttien välillä. Näin voidaan varmistaa, että lopullinen systeemi toteuttaa alkuperäiset sidosryhmävaatimukset. Auton käyttäjä haluaa tietyn kiihtyvyyden, jonka toteuttamiseksi auton suunnittelija määrittää tietyn moottorin tehon. Tämän perusteella moottorinohjausyksikön valmistajan täytyy toteuttaa oma tuotteensa niin, että koko systeemi toteuttaa lopulta alkuperäisen käyttäjävaatimuksen. [27]

2.7.5 Systeemivaatimukset

Systeemivaatimukset muodostetaan käyttäjien vaatimusten pohjalta. Ne kertovat miten systeemin tulee toimia, että se toteuttaa lopulta käyttäjien asettamat vaatimukset. INCOSE:n mukaan vaatimukset olisi järkevää jaotella neljään eri osioon: toiminnalliset vaatimukset, suoritusvaatimukset, ei-toiminnalliset vaatimukset ja arkkitehtuuriset rajoitukset.

Toiminnalliset vaatimukset kuvaavat, mitä systeemin tulisi pystyä tekemään sille tarkoitetulla toiminta-alueella ja määrittävät toiminnot, joita systeemin tulee suorittaa. Suoritusvaatimukset määrittävät, kuinka hyvin toiminnallisten vaatimusten tulee suoriutua. Ei-toiminnallisissa vaatimuksissa kuvataan se, minkälainen systeemi on. Lisäksi ei-toiminnallisiin vaatimuksiin voidaan lisätä vaatimuksia, jotka eivät suoraan liity systeemin toimintaan. Tällaisia voivat olla esimerkiksi systeemin turvallisuuteen ja käyttökoulutukseen liittyvät seikat. Arkkitehtuuriset rajoitukset ovat rajoja, joiden puitteissa muut vaatimukset toteuttavat systeemin ja ne otetaan huomioon myöhemmin arkkitehtuurin suunnittelussa. [21]

Systeemivaatimusten määrittelemisen jälkeen systeemivaatimukset verifioidaan sidosryhmillä ja tarkastetaan toteuttavatko ne alkuperäiset sidosryhmävaatimukset. Systeemivaatimukset dokumentoidaan ja pidetään huoli jäljitettävyydestä. Tämän jälkeen systeemivaatimukset voidaan välittää arkkitehtuurin, alisysteemien ja komponenttitason suunnitteluun.

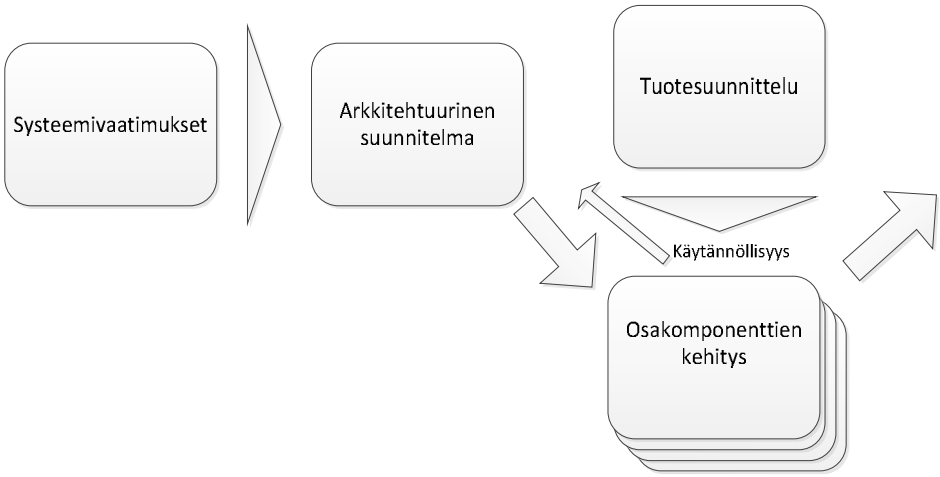
Vaatimusten määrittelyssä täytyy pitää kuitenkin huoli tietystä abstraktiotasosta. Systeemivaatimusten ei tule olla liian tarkkoja tulevan toteutuksen suhteen. Vaatimusten tarkkuus tulee olla sellainen, minkä perusteella systeemin arkkitehtuuri voidaan suunnitella sekä myöhemmin alisysteemien ja komponenttien suunnittelijat voivat toimia. Vaatimusten tulisi kuvata mitä tietyn osakokonaisuuden tulee suorittaa, eikä sitä kuinka se suoritetaan. [21]

2.7.6 Systeemin arkkitehtuurin suunnittelu

Tämän suunnitteluvaiheen tavoitteena on tuottaa suunnitelma systeemin arkkitehtuurista. Suunnitellun arkkitehtuurin tulisi toteuttaa sidosryhmävaatimusten perusteella määritetyt systeemivaatimukset. Tässä vaiheessa systeemisuunnitteluinsinöörit tekevät yhteistyötä systeemin toteutukseen ja osakomponenttien valmistukseen liittyvien alojen insinöörien kanssa. Arkkitehtuurin suunnitteluvaiheessa luodaan systeemin arkkitehtuurin perusteet, systeemin eri elementtien kuvaukset, erilaisten rajapintojen vaatimukset ja verifiointisuunnitelma. Lisäksi pidetään edelleen huoli, että vaatimusten jäljitettävyys säilyy. [21]

Systeemin arkkitehtuurisen suunnitelman täytyy kuvata systeemin rakenne, toiminta, ja layout. Rakenne kuvaa, mitä pääkomponentit ovat, niiden toiminnallisuuden sekä komponenttien väliset rajapinnat. Rakennekuvauksen pitää sisältää myös dokumentointi siitä, kuinka systeemin rakenne toteuttaa systeemivaatimukset. Systeemin toimintakuvaus on looginen malli, joka selvittää systeemin perustoiminnan ja kuinka systeemi käyttäytyy erilaisissa tilanteissa. Systeemin layout määrittää fyysiset järjestelyt ja kuinka eri komponentit jaetaan fyysisiin resursseihin. [27]

Arkkitehtuurisen suunnitelman perusteella voidaan toteuttaa systeemin alisysteemit ja edelleen erilaiset kokoonpano-osat. Suunnitelman tulisi olla abstraktiotasoltaan sellainen, että alisysteemien ja komponenttien valmistajilla on tarpeeksi suunnitteluvaraa. Kuvassa 7 on käsitteellinen kuvaus arkkitehtuurisen suunnitelman työvaiheista.

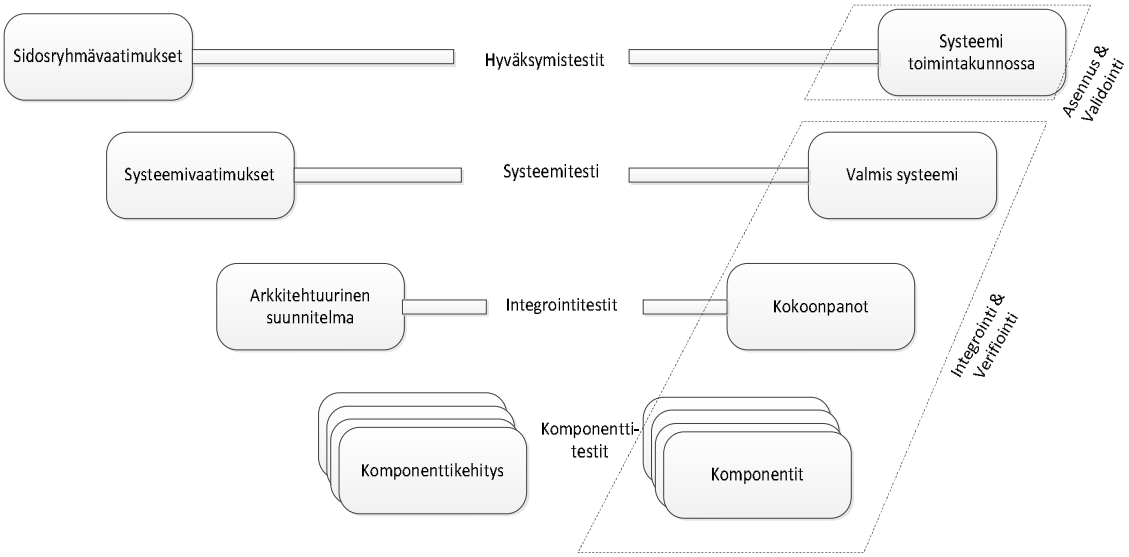


Kuva 8. *Arkkitehtuurinen suunnitelma, mukailtu lähteestä [27].*

2.7.7 Systeemin integrointi, verifiointi ja validointi

Systeemin komponenttien ja alisysteemien valmistus ei sinällään kuulu systeemisuunnittelijan työalueisiin. Osakomponenttien valmistuksen suorittaa kunkin erikoisalan osaajat ja valmistajat. Systeemisuunnitteluinsinöörin täytyy kuitenkin seurata, että valmistuvat komponentit noudattavat ennalta suunniteltua arkkitehtuuria ja vaatimuksia. [27]

Systeemisuunnittelun toiminnot liittyvät tässä vaiheessa siihen, että suoritetaan integrointi ja verifiointi systeemin alimmalta tasolta aina valmiin systeemin toimintaan asti. Lopuksi systeemi voidaan asentaa suunniteltuun toimintaympäristöön ja suorittaa systeemin validointi. [27] Kuvassa 8 on kuvattu yksittäisen systeemin integrointi, verifionti ja validointi V-mallissa.



Kuva 9. *Systeemin verifiointi ja validointi, mukaillen [27].*

Verifiointi olisi syytä suorittaa mahdollisimman alhaiselta tasolta ja mahdollisimman aikaisessa vaiheessa. Kun lopullinen systeemi on selvittänyt sille suoritetut testit, se voidaan asentaa sille suunniteltuun toimintaympäristöön. Systeemille voidaan nyt suorittaa validointi, jossa tarkastaa sopiiko lopullinen systeemi aiottuun käyttötarkoitukseen eli toteuttaako se sidosryhmien alkuperäiset vaatimukset. [27]

2.8 Systeemin uudelleensuunnittelu

Erittäin usein systeemisuunnittelun projekti on ainakin osaksi uudelleensuunnittelua ja kehittämistä vanhan systeemin pohjalta. Suunnittelussa vanhat ja toimivat alijärjestelmät sekä komponentit yhdistetään uusien kanssa. [25 s.83, 38]

Jo olemassa olevan systeemin uudelleensuunnittelu on yksi vaikeimmista tehtävistä systeemisuunnittelussa. Suunnitteluprosessi voi usein olla huomattavasti työläämpi kuin aivan uuden järjestelmän kohdalla. Näin on varsinkin silloin, jos aikaisempi systeemi on huonosti dokumentoitu. Usein kehitystä vaativat systeemit mahdollisesti toimivat, mutta ovat vanhentuneita teknologian osalta. Tämän myötä ne ovat usein kalliita ja vaikeita ylläpitää. Järjestelmät ovat voineet olla huonoja käyttää ja sisältää virheitä suunnittelussa, mutta sen käyttäjät ovat löytäneet keinot saada järjestelmä toimimaan. Yleensä virheet ja puutteet jäävät dokumentoimatta ja siksi uusien käyttäjien on vaikeaa omaksua vanhentuneiden systeemien toimintaa. [27]

Vanhat systeemit sisältävät usein paljon tietoa ja kokemusta, jonka ainoastaan sen käyttäjät ovat omaksuneet. Tämä tieto on usein dokumentoimatta. Siksi järjestelmää on vaikea kopioida sekä tehdä siihen päivityksiä ja muutoksia. Suuren uudistuksen jälkeen

uusi järjestelmä voi olla toiminnaltaan epäluotettava, koska vaatimuksia on jäänyt puuttumaan vanhan järjestelmän puutteellisen dokumentoinnin takia. [27]

Alkuperäisiä vaatimusmäärittelyjä ei usein ole tai ne ovat puutteellisia ja sekavia. Mahdollisissa vaatimusmäärittelyissä ei ole välttämättä jäljitettävyyttä. Myös muu suunnitteludokumentointi voi olla puutteellista.

Vaikka systeemin arkkitehtuuri ja toiminnallisuus tehdäänkin uudestaan, uuden systeemin vaatimuksia voidaan usein johtaa vanhan järjestelmän toiminnasta varsinkin samankaltaisten systeemien tapauksessa [38]. Myös vanhan järjestelmän heikkouksien havaitseminen on hyvä lähde vaatimusten määrittämiseen ja uuden järjestelmän parempaan toteutukseen. Systeemisuunnittelijan tehtävänä on päättää, mitkä osat systeemistä suunnitellaan uudelleen, mitkä säilytetään ja mitkä osat korvataan täysin uusilla. Myös uusien ja vanhojen toiminnallisuuksien keskinäisen yhteensopivuuden varmistaminen kustannusten, suorituskyvyn ja aikataulun perusteella on osa tärkeää suunnittelutyötä. [25 s.83]

Systeemin uudelleensuunnittelussa voidaan käyttää hyväksi myös mallipohjaista systeemisuunnittelua. Mallipohjaisten menetelmien avulla voidaan karakterisoida vanhasta systeemistä kuvaileva malli, jonka avulla uudelleensuunnittelu helpottuu. Toisaalta uuden systeemin malli voi toimia myöhemmin apuna systeemin kehittämisessä, koska systeemin elementtien ja vaatimusten uudelleenkäyttö helpottuu. Mallipohjaisesta systeemisuunnittelusta on kerrottu tarkemmin luvussa 3.

3. MALLIPOHJAINEN SYSTEEMISUUNNITTELU

Tässä luvussa selvitetään dokumentti- ja mallipohjaisen systeemisuunnittelun erot ja käydään läpi mallipohjaisen systeemisuunnittelun perusperiaatteet. Lopuksi tutustutaan mallipohjaisen systeemisuunnittelun tarpeisiin kehitettyyn SysML-kieleen sekä sen eri kaavioiden perusteisiin ja käyttötapoihin sillä tarkkuudella, että pystytään havainnoimaan tarvittaessa työssä muodostettavaa systeemin mallia.

3.1 Dokumenttipohjaisen systeemisuunnittelun lähestymistapa

Perinteistä systeemisuunnittelua kutsutaan usein dokumenttipohjaiseksi systeemisuunnitteluksi (*document-based systems engineering*). Suunnitteluprojektin aikana dokumentteja muodostuu esimerkiksi toimintakuvauksesta, vaatimusmäärittelyistä, insinööriraporteista ja testitapausten määrittelyistä. Näitä dokumentteja käytetään suunnitteluprosessin eri vaiheissa systeemin sidosryhmien, suunnittelijoiden ja testaajien kesken.

Dokumentit voivat olla esimerkiksi sanallisia selityksiä, taulukoita tai erilaisia suunnitteluasiakirjoja. [18,40] Dokumenttien ylläpito ajantasaisina on usein hankalaa ja kallista. Muutoksen tekeminen yhteen dokumenttiin aiheuttaa sen, että muutkin dokumentit, joita kyseinen muutos koskettaa täytyy päivittää. Tässä prosessissa tapahtuu usein virheitä esimerkiksi nimeämisissä. Muutoksia voi myös helposti unohtua tekemättä kaikkiin systeemin osa-alueisiin joihin muutos vaikuttaa. [18]

Jäljitettävyys suunnitellun arkkitehtuurin, alisysteemien, osien ja vaatimusten välille on dokumenttipohjaisessa systeemisuunnittelussa vaikea toteuttaa. Vaatimusten ylläpitoon käytetään usein jotain vaatimustenhallintaohjelmistoa. Ohjelmiston ja erilaisten suunnitteludokumenttien välinen yhteys on kuitenkin usein hankala toteuttaa automaattisesti ja muutos vaatimuksiin vaatii erikseen muutoksen sitä koskettavaiin dokumentteihin. Systeemivaatimusten uudelleenkäyttäminen kehittyvän järjestelmän tai vaihtoehtoisen toteutusvaihtoehtojen tapauksessa voi olla vaikeaa dokumenttipohjaisessa systeemisuunnittelussa. Nämä rajoitukset voivat aiheuttaa tehottomuutta ja laatuongelmia systeemin testauksessa tai pahimmillaan myös sen jälkeen, kun systeemi on toimitettu asiakkaalle. [40 s.16]

3.2 Mallipohjaisen systeemisuunnittelun lähestymistapa

Malli on epätäydellinen esitys todellisesta systeemistä. Se voi olla kuvaileva malli kuten esimerkiksi pienoismallit. Toisaalta se voi olla myös analyyttinen malli, kuten matemaattinen tai looginen malli, jonka avulla todellista systeemiä tai tilannetta voidaan ana-

lysoida ja simuloida. Analyyttiset mallit voidaan edelleen jakaa staattisiin ja dynaamisiin malleihin. Staattinen malli on ajasta riippumaton. Dynaaminen malli esittää systeemin ajasta riippuvaa tilaa. [40]

Mallipohjaista suunnittelua on käytetty kauan useilla insinöörialoilla esimerkiksi mekaniikka- ja sähkösuunnittelussa. Mallipohjainen systeemisuunnittelu eli MBSE (*model based systems engineering*) on suunnittelumenetelmä, jossa suunniteltavasta systeemistä muodostetaan kuvaileva malli. Delligatti määrittelee mallipohjaiseen systeemisuunnitteluun kolme peruspilaria: mallinnusmenetelmä, mallinnuskieli ja mallinnustyökalu [18]. Systeemin malli saadaan aikaan noudattamalla perinteisiä systeemisuunnittelun prosesseja ja käyttämällä mallipohjaisen systeemisuunnittelun menetelmiä. Malli luodaan aikaisessa vaiheessa suunnitteluprosessia ja se kehittyy projektin edetessä [22]. Mallin muodostamisessa voidaan käyttää esimerkiksi graafisia UML- tai SysML-mallinnuskieliä. Suunnittelua varten on olemassa useita kaupallisia sekä myös avoimia työkaluja.

Dokumenttipohjaiseen systeemisuunnitteluun verrattuna mallipohjaisessa systeemisuunnittelussa on useita etuja. Jokaista dokumenttia ei tarvitse päivittää erikseen suunnitteluvaiheen edetessä ja muutoksia tehdessä. Systeemi-insinöörit tekevät samanlaisen elinkaarisuunnittelun ja eri alojen suunnitteluinsinöörit tuottavat samat suunnittelutuotteet. Erillisten dokumenttien sijaan muodostetaan kuitenkin yhtenäinen systeemin malli MBSE-suunnittelutyökaluilla. Tarvittava informaatio systeemin suunnittelu-, testaus- ja käyttövaiheessa tuotetaan systeemin mallin pohjalta. [18,40]

Eräs suurimpia mallipohjaiseen systeemisuunnitteluun liittyviä etuja on, että jäljitettävyys on helpompi toteuttaa vaatimusten ja systeemin eri osien välillä. Systeemin mallin avulla voidaan varmistaa, että systeemi toteuttaa sille asetetut vaatimukset kaikissa suunnittelun eri vaiheissa. Kommunikaatio suunnitteluprojektin eri osapuolten välillä helpottuu, koska vaatimuksia voidaan tarkastaa helpommin sidosryhmien kanssa. Riski kehitysvaiheen epäonnistumisissa pienenee, koska vaatimuksia voidaan verrata jatkuvasti suunniteltuun. Tuottavuus paranee koska vaatimuksia voidaan lisätä ja suunnittelumuutoksia tehdä nopeammin. Suunnitteluvaiheessa pystytään tekemään tarkemmat kustannusarviot esimerkiksi käyttämällä apuna teknisiä mittareita. [18,40]

Mallipohjaisen systeemisuunnittelun perustana on yhtenäinen systeemin malli. Usein tarve voi kuitenkin olla erillisille dokumenteille. Useat mallipohjaisen systeemisuunnittelun ohjelmistot mahdollistavat dokumenttien luomisen automaattisesti mallin pohjalta. Systeemin mallista voidaan luoda erilaisia näkymiä, jotka muodostetaan niin, että ne palvelevat parhaiten kutakin sidosryhmää. Tämä lisää kustannussäästöjä entisestään.

Kun systeemin elinkaaren eri vaiheissa on saatavilla malli, käyttäjien koulutus helpottuu. Tiedonvälitys helpottuu, koska malli mahdollistaa tehokkaan pääsyn informaatioon ja sen muokkaamiseen. Mallin avulla voidaan demonstroida ja havainnollistaa systee-

min toimintaa. Erilaisia suunnitteluvaihtoehtoja voidaan vertailla helpommin ja vähemmin kustannuksin. Erilaiset testitapaukset voidaan määrittää osakomponenteille mallin avulla, joten integroinnin ja testauksen yhteydessä virheet vähenevät, eikä aikaa kulu niin paljon. Olemassa olevien järjestelmien suunnittelu on helpompaa, jos saatavilla on malli ja sitä voidaan käyttää hyväksi. Aiempia malleja, mallikirjastoja ja mallielementtejä voidaan käyttää hyväksi ja suunniteltujen osien uudelleen käyttäminen helpottuu huomattavasti. [18,40]

3.3 Systeemin malli

Mallipohjaisen systeemisuunnittelun menetelmillä luotu systeemin malli on kuvaileva malli. Systeemin malli kattaa systeemin rakenteen, vaatimukset, toiminnan sekä rajoitteet. Lisäksi systeemin malli selittää eri suunnitteluelementtien väliset suhteet ja rajapinnat. Mallista voidaan luoda erilaisia abstraktiotasoja systeemin eri suunnitteluvaiheiden aikana. Systeemin mallin avulla voidaan esittää erilaisia näkökulmia systeemistä eri sidosryhmille. [40]

Systeemin mallin lisäksi suunnitteluprojekti sisältää usein monia erilaisia osamalleja systeemin alisysteemeistä ja komponenteista. Osamallit voivat liittyä esimerkiksi sähkö- tai mekaniikkasuunnitteluun. Mallit ovat usein toteutettu erilaisilla mallinnustyökaluilla ja menetelmillä kuin systeemin malli. Mallipohjaisessa systeemisuunnittelussa systeemin mallin tulisi toimia yhdistävänä tekijänä erilaisten osamallien kesken.

Suuriin systeemeihin liittyvät analyyttiset mallit kuvaavat yleensä vain pienen osan systeemin toiminnallisuudesta, mutta fokusoituvat johonkin tiettyyn osaan ja ovat tarkempia. Mallipohjaisessa systeemisuunnittelussa systeemin malli ja erilaiset systeemiä tai sen osia kuvaavat analyyttiset mallit ovat päällekkäisiä. Tehtäessä muutoksia systeemin malliin tai analyyttisiin malleihin täytyy pitää huolta, että yhteneväisyys säilyy. [40] Mallin pohjalta muodostetut kaaviot ja automaattisesti luodut dokumentit ovat ainoastaan erilaisia näkymiä systeemin mallista, eivät osa mallia itseään [18].

Suunniteltavan systeemin mallin laajuus tulisi vastata aina tarkoitustaan. Laajutta voidaan kuvata kolmella ominaisuudella. Mallin leveys (*model breadth*) määrittää, kuinka suuren osan systeemistä mallin on syytä kuvata. Tätä systeemin mallin ominaisuutta täytyy pohtia varsinkin suurten systeemien tapauksessa, jossa koko systeemiä ei välttämättä tarvitse mallintaa projektin tarpeisiin. Jos olemassa olevaan systeemiin lisätään uutta toiminnallisuutta, voidaan mallintaa ainoastaan lisättävät osat. [40]

Mallin syvyyden (*model depth*) täytyy olla sopiva suunnittelun hierarkiatasoon suhteutettuna. Suunnittelun alkuvaiheessa ei tarvita usein hyvin tarkkaa kuvausta systeemin toiminnasta. Ensimmäisissä mallin versioissa riittää usein selittävä black box-malli. Myöhemmällä suunnittelutasolla mallilta vaaditaan enemmän yksityiskohtaisuutta. [40]

Mallin tarkkuus (*model fidelity*) kertoo, kuinka tarkasti mallin osat kuvaavat todellisuutta. Käyttäytymistä kuvaava mallin osa-alue voi olla yksinkertainen selitys tapahtumien kulusta, jos mallin tarvitsee olla ainoastaan havainnollistava. Jos vaaditaan, että mallia täytyy voida simuloida, vaaditaan huomattavasti tarkempia yksityiskohtia. [40]

3.3.1 Mallipohjaiset mittarit

Systeemisuunnittelussa voidaan käyttää erilaisia teknisiä mittareita suunnittelun ja johtamisen tukena. Näitä mittareita hyväksi käyttämällä voidaan vaikuttaa suunnittelun onnistumiseen sekä kustannusrajoissa ja aikataulussa pysymiseen. Mallipohjaisessa systeemisuunnittelussa mittarit voidaan sisällyttää systeemin malliin, jolloin niitä on helppo seurata ja muokata tarvittaessa suunnittelun edetessä. [40] Erilaisia mittareita voivat olla vaikuttavuuden mittarit MOE (*measures of effectiveness*), suorituskyvyn mittarit MOP (*measures of performance*) tai teknisen suorituskyvyn mittarit TPM (*technical performance measures*). [21,75]

Vaikuttavuuden mittarit (MOE) mittaavat toiminnallisia lukuja siitä, kuinka systeemi suoriutuu sidosryhmien toiminnallisista vaatimuksista. Erilaiset MOE:t määritetään systeemisuunnittelussa sidosryhmien vaatimusten määrittelyprosessin yhteydessä. Vaikuttavuuden mittareiden ei tulisi olla voimakkaasti korreloivia keskenään, jotta saadaan mahdollisimman laaja kuva systeemin suoriutumisesta. Selkein tapa on määrittää tehokkuuden mittarit lukuarvoina, mutta ne voivat olla myös kvalitatiivisia. [21,75]

Suorituskyvyn mittarit esittävät fyysisiä tai toiminnallisia ominaisuuksia. MOP mittaa kuinka systeemi täyttää sille asetetut suorituskykyvaatimukset. Nämä mittarit johdetaan vaikuttavuuden mittareiden pohjalta. Jokaista vaikuttavuuden mittaria vastaa yleensä useampi suorituskyvyn mittari. Suorituskyvyn mittarit määritetään systeemisuunnittelun vaatimusanalyysiprosessin yhteydessä, missä tapahtuu myös systeemivaatimusten määrittely. [21,75]

Teknisen suorituskyvyn mittarit mittaavat kuinka hyvin systeemin tai sen osan odotetaan täyttävän teknisen vaatimuksen. Nämä mittarit johdetaan usein suorituskyvyn mittareiden pohjalta. Yleensä yhtä teknisen suorituskyvyn mittaria vastaa ainakin yksi suorituskyvyn mittari. Teknisen suorituskyvyn mittareita kannattaa valita vain niille kriittisille parametreille, joiden alittuminen aiheuttaa riskejä projektin kustannuksiin, aikatauluun tai systeemin suorituskykyyn. Ilman teknisen suorituskyvyn mittareita systeemi voi valmistua aikataulussa ja sallittujen kustannusten rajoissa, muttei täytä keskeisiä vaatimuksia. [21,75]

3.4 Malliperustainen arkkitehtuuri

UML-kieli ja sen profiilina myös SysML-kieli perustuu OMG yhtymän luomaan malliperustaisen arkkitehtuurin MDA-standardiin (*model driven architecture*). Malliperustai-

nen arkkitehtuuri luotiin ohjelmistotuotannon tarpeita ajatellen. Se tarjoaa keinot systeemin ymmärtämiseen, suunnitteluun, valmistamiseen, toimintaan, kehittämiseen, ylläpitoon ja muutoksiin. Malliperustaisen arkkitehtuurin päätavoitteet ovat: siirrettävyys, yhteensopivuus ja uudelleenkäytettävyys [48].

Malliperustaisen arkkitehtuurin perusajatus on, että voidaan luoda alustariippumaton malli suunnitteluprojektia varten. Arkkitehtuurin käyttö suunnitteluprojekteissa mahdollistaa systeemin mallin siirtämisen eri suunnittelualustoille [48]. Malliperusteisessa arkkitehtuurissa systeemiä voidaan tarkastella laskentariippumattomasta, alustariippumattomasta tai alustariippuvasta näkökulmasta.

Laskentariippumaton malli CIM (*computation independent model*) on näkymä järjestelmästä laskentariippumattomasta näkökulmasta. Mallissa ei näytetä systeemin tarkkaa rakennetta. Rakenteen yksityiskohdat ja toiminta ovat piilotettu tai ne ovat määrittelemättä. Laskentariippumatonta mallia kutsutaan usein toimialamalliksi tai liiketoimintamalliksi. Systeemin vaatimusten kuvaus voidaan katsoa laskentariippumattomaksi malliksi. [48]

Alustariippumaton malli PIM (*platform independent model*) on näkymä systeemistä alustariippumattomasta näkökulmasta. Alustariippumaton näkökulma keskittyy esittämään systeemin toimintaa piilottamalla yksityiskohdat, jotka voitaisiin esittää ainoastaan jollain tietyllä alustalla. Alustariippumaton malli voidaan muodostaa käyttämällä jotain yleisluontoista mallinnuskieltä tai kieltä, joka on tarkoitettu toteutettavan systeemin toteutusalalle. [48].

Alustariippuva malli PSM (*platform spesific model*) on näkymä järjestelmästä alustariippuvasta näkökulmasta. Malli yhdistää alustariippumattoman mallin määrittelyihin yksityiskohtia, joita systeemi käyttää tietyllä alustalla. [48]

SysML-kieli perustuu myös OMG:n MOF-mallikehykseen (*Meta-Object Facility*). Mallikehys kuvaa mallin tietoja ja kuinka eri mallielementit ovat yhteydessä toisiinsa. MOF mallikehys perustuu malliperustaiseen arkkitehtuuriin ja sen tarkoituksena on mallien alustariippumattomuus. SysML-kielen mallien siirto alustojen välinen tapahtuu XMI-teknologian (*XML metadata interchange*) avulla. XMI määrittää, kuinka UML-malli ja sen profiilien mallit, kuten SysML-mallit kuvautuvat XML-kielelle. [40,49]

3.5 UML

UML eli unified modelling language on graafinen mallinnuskieli, joka on kehitetty alun perin ohjelmistokehitystä varten. Varsinkin oliopohjainen ohjelmistokehitys oli vahvasti mukana mallinnuskielen synnyssä. Kieli muodostui kolmen johtavan (Booch, OMT, OOSE) oliopohjaisen työtavan pohjalta [47]. Kieli on standardisoitu ohjelmistotuotannon standardien ylläpitoon keskittyvän yhteenliittymän object management groupin

(OMG) toimesta vuonna 1997. UML-kieltä on käytetty ohjelmistosuunnittelun lisäksi esimerkiksi liiketoimintaprosessien mallinnuksessa sekä mallipohjaisessa systeemisuunnittelussa.

UML sisältää erilaisia kaaviotyyppejä, joita käyttäjä voi hyödyntää suunnitteluprojektissaan. Pääkaaviotyypit ovat: käyttäytymiskaavio, vuorovaikutuskaavio ja rakennekaavio. Erilaisia kaavioita voidaan käyttää ohjelmiston rakenteen suunnitteluun, jonka pohjalta itse ohjelmointityö on johdonmukaista toteuttaa.

Fowler jakaa UML käyttäjät kolmeen ryhmään [41 s.2]. Ensimmäinen ryhmä ovat käyttäjät, jotka käyttävät UML-kieltä järjestelmän luonnostelussa. Luonnos on ainoastaan hahmotelma toteutettavasta ohjelmasta ja sitä voidaan käyttää kommunikoinnin apuna työntekijöiden kesken. Luonnos voidaan tehdä myös olemassa olevan ohjelmakoodin ymmärtämisen tueksi. [41]

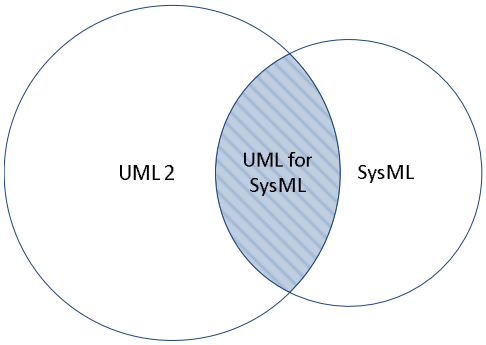
Toinen ryhmä ovat käyttäjät, jotka tekevät tarkan suunnitelman ohjelmointityöstä UML-kielen avulla. Tällöin tarkkuus täytyy olla sellainen, että suunnitelman perusteella ulkopuolinen ohjelmoija voi toteuttaa suunnitelman. [41]

Kolmas ryhmä käyttää UML-kieltä ohjelmointikielenä. Tällöin UML-kaaviot käännetään suoritettavaksi ohjelmakoodiksi. Tässä tapauksessa vaaditaan usein kehittyneempiä työkaluja. [41]

3.6 SysML

SysML on suhteellisen nuori mallinnuskieli, joka kehitettiin mallipohjaisen systeemisuunnittelun tarpeisiin UML-kielen pohjalta. UML-kieltä oli käytetty mallipohjaisessa systeemisuunnittelussa pitkään, mutta sitä pidettiin kuitenkin liian ohjelmistokeskeisenä. Lisäksi kielessä oli selviä puutteita, joita tarvittiin mallinnettaessa kompleksisia systeemejä. Vuonna 2001 kansainvälinen systeemisuunnittelun organisaatio INCOSE yhdessä OMG-yhteenliittymän kanssa asetti tavoitteekseen luoda UML-kielen pohjalta standardikielen systeemisuunnitteluun [23]. UML-kieltä laajennettiin useilla elementeillä kuten mahdollisuudella mallintaa helpommin vaatimuksia sekä jatkuvia järjestelmiä. SysML-kielen ensimmäinen versio ilmestyi vuonna 2006 ja on vakiintunut alan standardiksi mallinnettaessa kompleksisia systeemejä, jotka sisältävät laitteistoa, ohjelmistoja, ihmisiä, toimintatapoja sekä palveluita. Kieli mahdollistaa vaatimusten, rakenteen, käyttäytymisen ja parametrien mallintamisen siten, että voidaan muodostaa selkeä kuvaus systeemistä, alikomponenteista ja toimintaympäristöstä. [40]

SysML-kielen lähtökohta on vaatimuslähtöinen. Mallinnuskieleen on tehty tiettyjä lisäyksiä UML:ään nähden, että voidaan täyttää systeemisuunnittelun tarpeet. SysML koostuu UML-kielen osajoukosta ja erillisestä profiilista. SysML-kielessä UML-kielestä on uudelleenkäytetty vajaa puolet, joka selviää kuvasta 10.



Kuva 10. *UML ja SysML yhteiset osa, mukaillen [23].*

Friedenthal [40] luettelee seuraavat viisi systeemin näkökantaa, joita SysML-kielen avulla voidaan esittää:

- Rakenteellista koostumusta, yhteyksiä ja luokittelua
- Toiminnallista, viestipohjaista, ja tilamuotoista käyttäytymistä
- Rajoituksia systeemin fysiikkaan ja suorituskykyyn liittyen
- Jaottelua käyttäytymisen, rakenteen ja rajoituksien välillä
- Vaatimuksia sekä niiden suhteita muihin vaatimuksiin ja suunnitteluelementteihin

Mallinnuskieli määrittelee minkä osatekijöiden perusteella mallin voi rakentaa. SysML kielen lisäksi systeemisuunnitteluprosessiin liittyy usein monia insinöörialoja, ja niiden osaajat käyttävät muitakin graafisia mallinnuskieliä kuten UML, IDEF0, MARTE tai AADL [45,46]. Erilaisten systeemien mallinnukseen on olemassa myös tekstipohjaisia mallinnuskieliä. Analyyttisten mallien simulointiin tarkoitettu Modelica on oma tekstipohjainen mallinnuskielensä. [18]

3.7 MBSE mallinnustyökalut

Mallinnustyökalu on ohjelmisto, jolla systeemin malli voidaan muodostaa mallinnuskieltä käyttämällä. Jotkin mallinnustyökalut voivat tukea useampaa erilaista mallinnuskieltä. Eräitä käytetyimpiä kaupallisia mallinnustyökaluja ovat Cameo Systems Modeler, MagicDraw, IBM Rhapsody ja Enterprise Architect. Ilmaisia mallinnustyökaluja ovat esimerkiksi Modelio ja Papyrus.

Mallinnustyökalun valinnassa tulisi miettiä ja arvioida tarvetta ainakin seuraaville asioille:

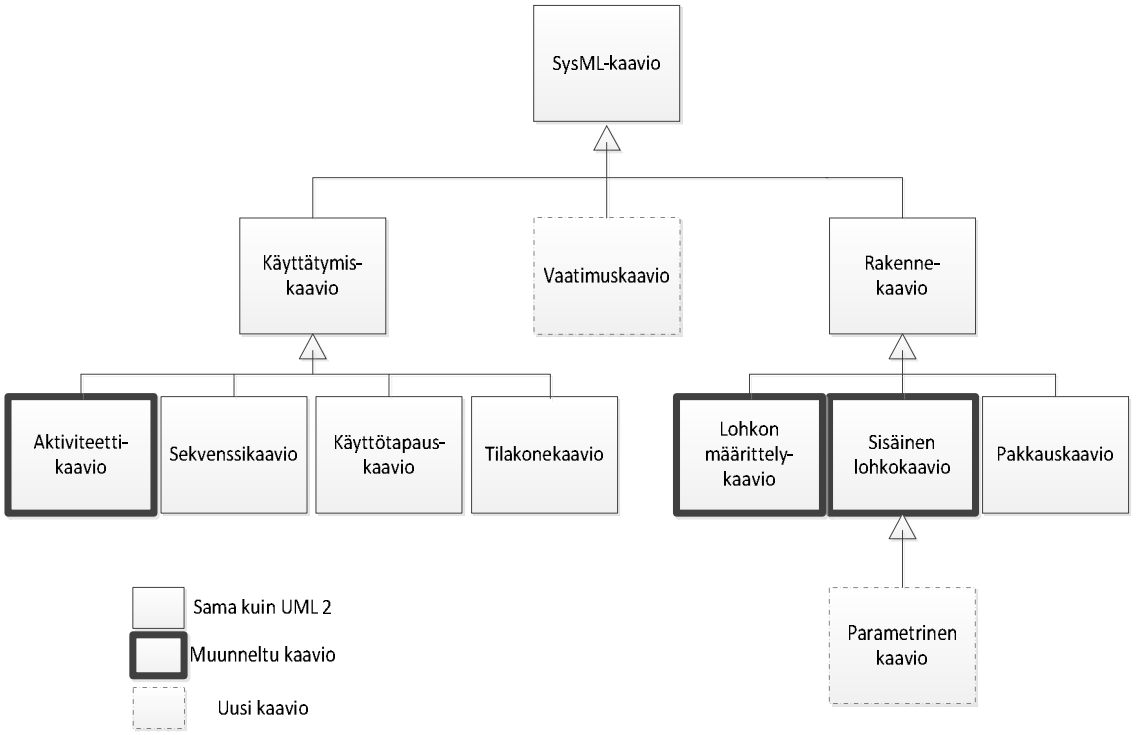
- yhteensopivuus mallinnuskielen viimeisimmän version kanssa

- käytettävyys
- dokumenttien ja raporttien automaattinen generointi
- alustariippumattomien mallien muodostaminen
- mahdollisuus mallirepositorion käyttöön
- yhteensopivuus muiden suunnittelutyökalujen kanssa [40]

Simuloinnin mahdollisuus on ollut pitkään analyyttisten mallien osana. Mallipohjaisen systeemisuunnittelun myötä systeemin mallin osien simulointi on tullut ajankohtaiseksi. Simulointi koskee yleensä vain toiminnallisia osia mallista. SysML-kielen tapauksessa simulointi joudutaan usein tekemään jollain ulkopuolisella ohjelmalla tai kehitetty malli täytyy kääntää toiselle kielelle, jos mallinnukseen käytetty kehitysympäristö ei tarjoa mahdollisuutta mallin osien simulointiin. Esimerkiksi SysML-kieleen on kehitetty Modelica-laajennus. Kyseisen laajennuksen avulla voidaan luoda SysML-malli, jonka pohjalta voidaan muodostaa Modelica-ohjelmakoodi. [44].

3.8 SysML-kielen kaaviot

SysML-kielessä systeemin mallia kuvataan erilaisten kaavioiden avulla. Koska SysML on UML-kielen osajoukko ja profiili, ovat mallintamiseen käytettävät kaaviot osaksi samat kuin UML-kielessä. Käyttäytymistä kuvaavia kaaviotyyppejä on SysML-kielessä neljä kappaletta: aktiviteettikaavio, sekvenssikaavio, käyttötapauskaavio ja tilakonekaavio. Rakennetta voidaan kuvata myös neljällä erilaisella kaaviotyypillä. Rakennekaavioita ovat lohkon määrittelykaavio, pakkauskaavio ja sisäinen lohkokaavio. Lisäksi SysML-kieleen on lisätty parametrinen kaavio, joka on sisäisen lohkokaavion erikoistapaus. Vaatimusten mallintamista varten SysML-profiiliin on lisätty erityinen vaatimuskaavio. Erilaiset SysML-kaaviot ja niiden yhtäläisyydet UML-kaavioiden kanssa ovat esitetty kuvassa 11.



Kuva 11. *SysML-kaaviotyypit, mukaillen [50].*

3.8.1 Lohkon määrittelykaavio

Lohko (*block*) on SysML-kielessä rakenteen perusyksikkö. Lohkolla voidaan kuvata mitä tahansa kokonaisuutta systeemissä tai sen ulkopuolella. Lohkossa on oltava ainakin nimiosio. Lisäksi lohkoille voidaan kuvata erilaisia ominaisuuksia siihen sisältyvissä lokeroissa. Erilaisia ominaisuuksia kuvaavia lokeroita lohkossa ovat: parts, referenecs, values, constraints, operations, recpetions, flow properties, structure. Lisäksi lohkolle voidaan määrittää esimerkiksi rajapintoja porttien avulla.

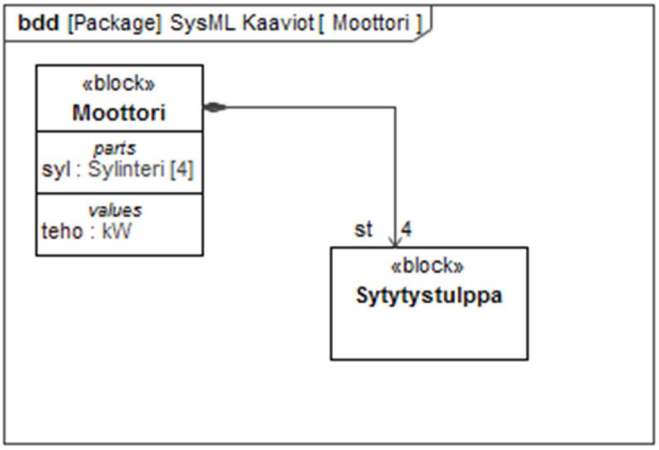
Lohkon määrittelykaavio (*block definition diagram*) kuvaa lohkojen rakenteelliset osiot sekä niiden kokoonpanon. Lohkon määrittelykaavio on johdettu UML-kielen luokkakaavion pohjalta. Kaavion lyhenne SysML-kielessä on **bdd** ja mallielementtinä se voi esittää pakkausta, mallia, mallikirjastoa, näkymää, lohkoa tai rajoitelohkoa.

Lohkon osaominaisuudet voidaan kuvata lohkon *parts*-osiossa. Osaominaisuudet ovat joitain muita lohkoja, jotka mallinnettava lohko omistaa. Tämä omistajuus tarkoittaa hiukan eri asiaa riippuen siitä minkälaista kokonaisuutta lohko kuvaa. Fyysisten kokonaisuuksien tapauksessa lohko koostuu fyysisistä osista. Ohjelmistojen tapauksessa omistajuus viittaa mallinnettavan kohteen velvollisuudesta objektien luomisessa ja tuhoamisessa. Kun muistia jaetaan kootulle objektille, sitä jaetaan samalla sen osille. Samoin muistin vapautuksessa vapautetaan muisti myös osaobjekteilta. [18] SysML-kielessä omistajuus merkitsee lisäksi sitä, että tietty osa voi kuulua vain yhteen kokonai-

suuteen kerrallaan. Osaominaisuuteen voidaan asettaa oletusarvo kerrannaisuudesta, eli kuinka monta osaa lohko omistaa, kuten kuvan 12 esimerkissä.

Osaominaisuudet voidaan kuvata lohkon määrittelykaaviossa myös koostumussuhteen avulla. Suhde ilmaistaan nuolella, jossa omistajan päässä on väritetty ruutukuvio. Nuolenkärki osoittaa osan suuntaan. Osan päässä voidaan esittää osan nimi sekä mahdollinen kerrannaisuus. Jos osaominaisuus on esitetty koostumussuhteen avulla, sitä ei yleensä merkitä erikseen omistavan lohkon osaominaisuuslokeroon. [40]

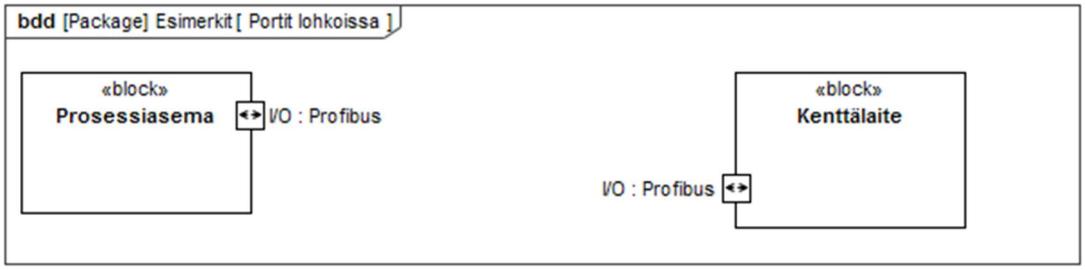
Lohkon arvo-ominaisuudet ovat listattu lohkon *values*-osiossa. Arvo-ominaisuus voi esittää lukua, tosiarvoa tai merkkijonoa. Yleisimmin arvo-ominaisuus on jotain, jonka voi kuvata lukuarvolla. Kerrannaisuus on rajoite siitä, kuinka monta tiettyä arvoa arvo-ominaisuus voi omistaa. Oletusarvo on valinnainen kohta, joka määrää minkä lukuarvon arvo-ominaisuus saa objektia luodessa, jos oletusarvo on asetettu. Arvo-ominaisuudesta lohkossa on esimerkki kuvassa 12.



Kuva 12. *SysML-lohkon osa- ja arvolokerot.*

Porteilla voidaan mallintaa kaikenlaisia yhteyspisteitä. Se voi mallintaa fyysistä objektia laitteen rajapinnassa, kuten esimerkiksi jotain mittaria. Se voi olla vaikutuspiste ohjelmisto-objektin rajapinnassa, esimerkiksi graafinen käyttöliittymä. Yhteyspisteellä voidaan mallintaa myös esimerkiksi kahden liiketoimintaorganisaation välinen vaikutus kuten osto. SysML ei määrittele rajoituksia sen suhteen mitä portti voi mallintaa.

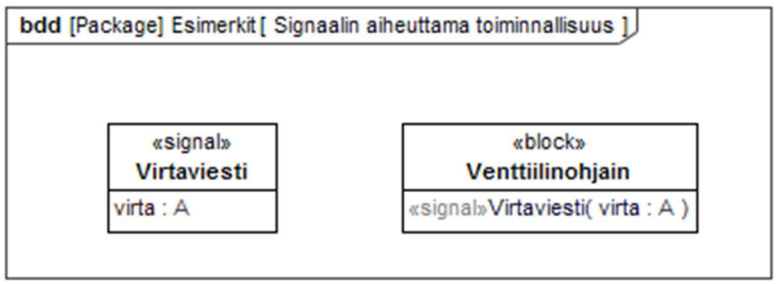
SysML 1.2 ja sitä aikaisemmat versiot määrittävät kaksi porttityyppiä. Standardiportin (standard port) ja virtausportin (flow port). Versiosta 1.3 alkaen SysML ei enää tue edellä mainittuja porttityyppejä, vaan määrittelee uudet *full port* ja *proxy port* porttityypit. [18,50] Tässä työssä mallinnettaessa käytetään edelleen SysML1.2 version porttityyppejä, koska sitä käytetään edelleen laajasti. Lisäksi kirjallisuuden esimerkkimallit käyttävät vielä yleisesti SysML 1.2 version porttityyppejä [18,40]. Kuvassa 13 on esimerkki virtausporteista, jotka kuvaavat kenttäväyläliityntää.



Kuva 13. *Porttien käyttäminen lohkojen rajapintoina.*

Lohkojen käyttäytymistä kuvaavia ominaisuuksia voidaan määritellä kahdella eri tavalla. Operaatiot osiossa (*operations*) määritellään lohkon käyttäytyminen kutsutilanteessa. Yleisimmin operaatio esittää synkronista käyttäytymistä. SysML-kielessä ei ole kuitenkaan vaatimusta tästä, vaan kyse voi olla myös asynkronisesta käyttäytymisestä.

Reaktiot (*receptions*) ovat käyttäytymistä, jonka lohko suorittaa saadessaan signaalin. Käyttäytymisen laukaiseva signaali on itsessään mallielementti. Lohko on hyväksytty signaalin kohde, jos se omistaa samannimisen reaktion kuin mitä laukaiseva signaali on. Reaktion nimeä täytyy edeltää avainsana <<signal>>. Lisäksi parametrien täytyy olla yhteneväiset. Reaktion aiheuttama käyttäytyminen on aina asynkronista. Signaalista ja lohkon sisältämästä reaktiosta on esimerkki kuvassa 14.

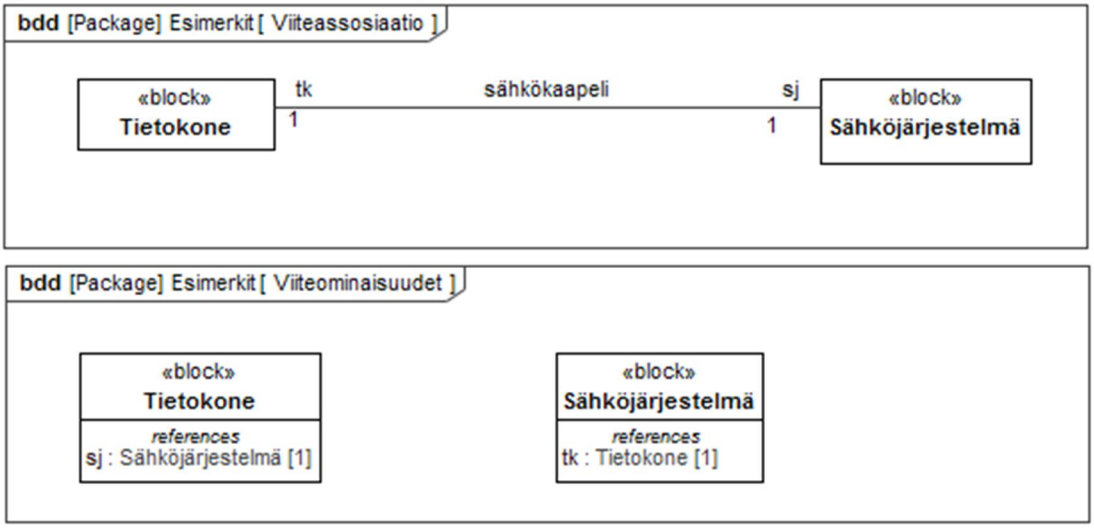


Kuva 14. *Signaali ja signaalin aiheuttama reaktio lohkossa.*

Rakenneominaisuudet voidaan määritellä rakenneosiossa (*structure*). Siinä lohkon sisäinen rakenne voidaan määritellä samalla tapaa kuin sisäisen lohkokaavion tapauksessa, josta lisää luvussa 3.8.2. Rakenneosiota harvemmin käytetään mallinnuksessa, koska rakennetta on usein selkeämpi esittää erillisessä kaaviossa. [18,50]

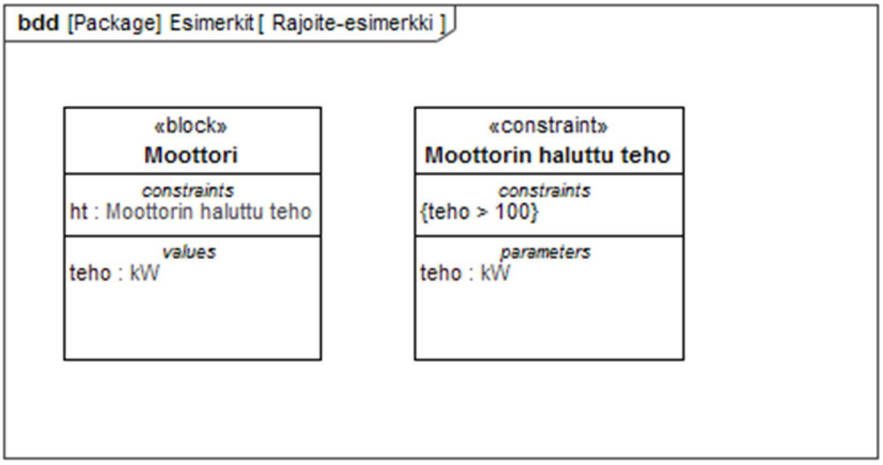
Lohkojen viiteominaisuudet (*references*) eivät tarkoita omistajuutta, kuten osien tapauksessa. Viiteominaisuus osoittaa enemmänkin tarvetta eri elementtien välillä. Viiteominaisuuden tyyppi täytyy kuitenkin olla jokin lohko tai toimija, joka on määritelty muualla systeemin mallissa. Viiteominaisuuden yhteydessä voidaan määritellä kerrannaisuus, joka on rajoite sille kuinka monta instanssia viitteestä voidaan luoda. Viiteominaisuus voidaan merkitä myös assosiaatioviivalla lohkojen välille. Viite voi olla yksi- tai kak-

sisuuntainen. Yksisuuntaisuus ilmaistaan avoimella nuolenkärjellä. [40] Viiteassosiaatiot on esitetty kuvassa 15.



Kuva 15. *Viiteassosiaatiot, mukaillen[18].*

Lohkon rajoiteominaisuudet kuvataan lohkon constraints-osiossa. Rajoitusominaisuus kuvaa yleensä matemaattisella relaatiolla vaatimusta, joka on tarkoitettu arvo-ominaisuuksia varten. Rajoitusta varten kannattaa yleensä luoda erillinen rajoituslohko, joka on esitetty kuvassa 16. Toinen tapa on määrittää rajoitusta kuvaava matemaattinen yhtälö kaarisulkeissa suoraan lohkon rajoitusosiossa. Tätä tapaa voi käyttää esimerkiksi silloin, jos rajoitusta käytetään vain kerran. [18]

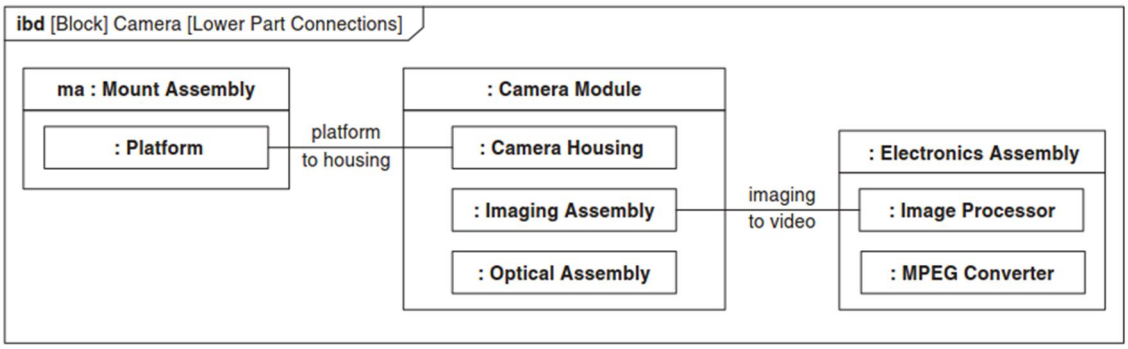


Kuva 16. *Rajoitusominaisuus ja erillinen rajoitelohko.*

Rajoiteominaisuudet ovat olennainen osa myöhemmin tässä luvussa esiteltävän parametrisen kaavion sisältöä.

3.8.2 Sisäinen lohkokaavio

Sisäinen lohkokaavio (*internal block diagram*) kuvaa lohkon sisäisten rajapintojen yhteyksiä. Sen käyttötarkoitus on lähellä lohkon määrittelykaaviota. Sisäisen lohkokaavion avulla voidaan tarkentaa esimerkiksi lohkon määrittelykaaviossa kuvatun lohkon sisäistä rakennetta tarkemmin. Lohko on ainoa mallielementti, jota sisäinen lohkokaavio voi kuvata. Lohkon määrittelykaaviossa voidaan mallintaa lohkon osaominaisuuksia ja viiteominaisuuksia lohkon osioissa sanallisesti. Sisäisen lohkokaavion avulla voidaan selventää näitä yhteyksiä eri osaominaisuuksien välillä. Esimerkki sisäisen lohkokaaviosta, joka esittää lohkon osien yhteyksiä on kuvassa 17. Sisäinen lohkokaavio luodaan usein samalla kuin lohkon määrittelykaavio. [18]



Kuva 17. *Sisäinen lohkokaavio [40].*

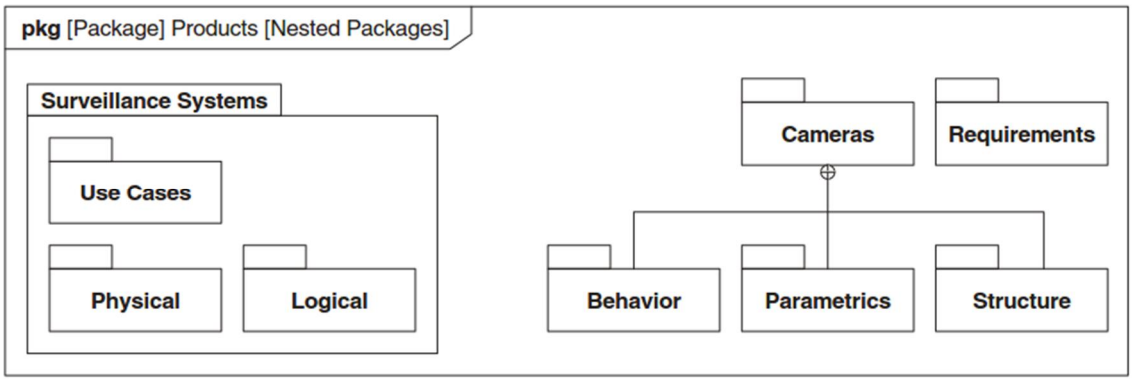
Vaikka sisäisen lohkokaavion avulla voidaan kuvata jonkin lohkon loogista sisäistä rakennetta, ei se mahdollista rakenne-elementtien tarkkaa geometrista kuvausta. Tämän tyyppinen suunnittelutyö joudutaan tekemään jollain ulkopuolisella mallinnusmenetelmällä. Sisäinen lohkokaavio mahdollistaa ainoastaan lohkon eri osien kuvaamisen sekä niiden välisten yhteyksien selittämiseen. [18]

3.8.3 Pakkauskaavio

Pakkausten tarkoitus on organisoida SysML-mallin elementtejä ja kaavioita suuremmiksi kokonaisuuksiksi. Pakkauskaavion avulla voidaan esittää erilaisten pakkausten sisältöä mallin rakenteessa. [40 s.109] Pakkaukset toimivat SysML-kielessä säiliöinä ja nimiavaruuksina. Jokaisella systeemin mallin elementillä on paikka säiliöhierarkiassa. Sen lisäksi jokainen elementti kuuluu nimiavaruuteen. Nimiavaruus mahdollistaa mallin jokaisen elementin yksikäsitteisen tunnistamisen. Jos elementtiä käytetään sen nimiavaruuden ulkopuolella, se voidaan paikantaa tarkalla nimellä.

Pakkauksen sisältämä nimiavaruus voidaan kuvata kolmella eri tavalla. Ensimmäinen tapa on tähtäinnotaatiolla, joka on esitetty kuvassa 18. Toinen mahdollisuus on pakkausten piirtäminen sisäkkäin, jolloin sisällä olevat pakkaukset kuuluvat uloimman pakkauksen nimiavaruuteen. Kolmas tapa on merkitä se tarkalla nimellä. Esimerkiksi kuvan

18 Cameras-pakkauksen sisältämä Structure-pakkaus voitaisiin merkitä pakkaussymbolilla ja tarkalla nimellä **Cameras::Structure.** [18,40]



Kuva 18. *Pakkauskaavio [40].*

Pakkauskaavio voi esittää mallielementtinä pakkausta, mallia, mallikirjastoa, näkymää tai profiilia. SysML-kielessä on tavallisen pakkauksen lisäksi neljä erikoistunutta pakkaustyyppiä:

- Malli (*model*)
- Mallikirjasto (*model library*)
- Profiili (*profile*)
- Näkymä (*view*)

Malli on erikoistunut pakkaustyyppi, joka toimii säiliöhierarkian juurena. Toisin sanoen malli on ylimmän tason pakkaus. Mallin esitystapa on samanlainen kansiosymboli kuin muilla pakkaustyypeillä, mutta nimen edessä on avainsana <<model>>. Toinen vaihtoehto esittää malli on sisällyttää kolmiosymboli kansion oikeaan ylänurkkaan merkitsemään kyseisen pakkauksen olevan malli. [18]

Mallikirjasto on pakkaustyyppi, joka sisältää elementtejä, joita tullaan käyttämään useissa malleissa tai mallin osissa. Mallikirjaston notaatio on sama kuin pakkauksella eli kansio, mutta nimen yläpuolella on avainsana <<modelLibrary>>. Mallikirjastot voivat sisältää esimerkiksi erilaisia arvotyyppejä tai rajoituslohkoja. Ne voivat koostua myös lohkoista, joita tarvitaan usein esimerkiksi laitteistojen ja ohjelmistojen määrittämisessä. [18]

Profiili on pakkaus, joka sisältää stereotyyppejä. Stereotyyppi määrittää uudenlaisen mallielementin lisäämällä rajoituksia tai merkityksiä jo olemassa olevalle mallielementille. Esimerkiksi SysML on osaksi UML-kielen profiili.

Näkymä on pakkaustyyppi, jonka avulla voidaan kuvata malli osittain esimerkiksi tietyn sidosryhmän tarpeita ajatellen. Näkymä tuo esille muita pakkauksia, elementtejä ja kaavioita. Näkymä toteuttaa aina tietyn näkökulman. Näkökulma (*viewpoint*) on mallielementti, joka määrittelee näkymän muodostamiseen ja näkymän käyttöön liittyvät

perusasiat. Näkökulma esitetään nelikulmiolla, missä nimiosiossa on avainsana <<viewpoint>> ennen nimeä. Näkökulma sisältää viisi ominaisuutta, jotka kuvaavat näkymän muodostamista ja sen käyttötarkoitusta:

- Stakeholders
- Concerns
- Purpose
- Languages
- Methods

Stakeholders ominaisuus kertoo mitä sidosryhmiä varten näkökulma on tarkoitettu. Concerns kertoo mihin sidosryhmien kysymyksiin näkökulma antaa vastaukset. Purpose kertoo sanallisesti, miksi tekijä on määrittänyt näkökulman. Languages ominaisuus kertoo mitä mallinnuskieliä käytetään. Methods ominaisuus kertoo minkälaisten sääntöjen perusteella mallintaja muodostaa näkökulman.

Erilaisia pakkauksia voidaan sisällyttää toisiin pakkauksiin import-suhteen avulla. Usein mallintaja tuo malleihin profiileja ja mallikirjastoja varsinkin mallinnuksen alkuvaiheessa.

3.8.4 Käyttötapauskaavio

Käyttötapaukset (*use case*) ovat kuvauksia systeemin tai sen osan toiminnallisuudesta. Ne mallintavat tyypillisiä tapahtumia systeemin käyttäjien ja itse systeemin välillä [41]. Jokainen systeemin käyttäytymistä tai toiminnallisuutta kuvaavaa tapahtuma ei ole kuitenkaan käyttötapaus. Käyttötapaukseen liittyy ulkoisen toimijan vaikutus systeemiin. [18]

Käyttötapaukset voivat olla pelkästään tekstipohjaisia kuvauksia käyttäjien suorittamista tapahtumaketjuista systeemin toiminnan mallintamisessa, josta on esimerkki taulukossa 2. Yleinen tapa nykyään on tehdä käyttötapauksien mallintaminen lisäksi graafisen kaavion avulla varsinkin mallipohjaisten menetelmien yhteydessä. Systeemin mallin suunnittelussa ensimmäiset käyttötapaukset määritellään yleensä aikaisessa vaiheessa [18].

Käyttötapaukset soveltuvat hyvin toiminnallisten vaatimusten mallintamiseen systeemissä [41]. Systeemin kehityksen edistyessä vaatimusten toteutumista voi seurata käyttötapausten avulla. Fowler ehdottaa käyttötapausten määrittelyyn käytettäväksi ensin toiminnan kuvaavia skenaarioita, joiden perusteella luodaan itse käyttötapaus. Skenaario on spesifinen tapahtumaketju, jonka käyttötapaus voi kuvata. [41] Käyttötapauksen tarkkuus on silloin sopiva, kun sillä on ainoastaan yksi onnistunut pääskenaario (*main success scenario*) [51].

Käyttötapauksista puhuttaessa systeemin käyttäjiä kutsutaan toimijoiksi (*actor*). Nämä toimijat voivat olla joko ihmisiä tai ulkoisia systeemejä, jotka vuorovaikuttavat käyttötapauksessa kuvattavan systeemin kanssa [18 s.78, 40 s.303].

Taulukko 2. *Esimerkki käyttötapauskuvauksesta.*

Nimi:	Ilmoittaudu kurssille
Päätoimija:	Opiskelija
Laajuus:	Ilmoittautumisjärjestelmä
Sidosryhmät:	Opiskelija, kurssin vastuuhenkilö
Esiehto:	Opiskelijalla on opinto-oikeus
Onnistunut pääskenaario:	1. Opiskelija kirjautuu järjestelmään 2. Opiskelija ilmoittautuu kurssille 3. Järjestelmä ilmoittaa onnistuneesta ilmoittautumisesta 4. Kurssin vastuuhenkilö saa ilmoituksen 5. Opiskelija kirjautuu ulos järjestelmästä
Poikkeukset:	1 a. Järjestelmän palvelin on kaatunut 2 a. Opiskelijalla ei esitietovaatimuksia

Käyttötapauskaavio SysML-kielessä koostuu itse käyttötapauksista, toimijoista ja niitä yhdistävistä assosiaatioista. Sallitut mallielementtityypit, joita käyttötapauskaavio voi esittää, ovat pakkaus ja lohko. Itse käyttötapaukset kuvataan kaaviossa ovaaleilla, joiden sisällä on käyttötapauksen nimi. Yleensä käyttötapauksen nimenä on verbi, koska käyttötapaus kuvaa toiminnallisuutta. Toimijat ovat järjestelmän käyttäjiä. Ne voivat vaikuttaa systeemin toimintaan suoraan tai toisen käyttäjän kautta [40]. Toimijaa kuvataan yleensä käyttötapauskaaviossa tikku-ukolla. Toinen vaihtoehto toimijan kuvaamiseen on nelikulmio avainsanan <<actor>> kanssa. [18 s.83,40 s.304]

Käyttötapauskaaviossa voidaan esittää myös systeemiraja (*system boundary*) nelikulmiolla, joka sisältää käyttötapaukset. Systeemirajaa ei pidä kuitenkaan sekoittaa itse kaavion reunoihin. Toimijat sijoitetaan kaaviossa systeemirajan ulkopuolelle, koska ne ovat kuvattavan systeemin ulkopuolisia toimijoita. Alkuvaiheessa systeemin konseptia suunniteltaessa systeemirajat voivat sisältää koko systeemin ja tällöin sen nimenä on suunniteltavan systeemin nimi. Myöhemmässä vaiheessa tästä koko systeemistä erotetaan pienempiin osiin suunniteltavia alisysteemejä ja komponentteja. Tällöin niiden systeemirajan nimi on kyseinen alisysteemi tai komponentti. [18] Kuvassa 19 on esitetty systeemiraja, joka esittää valvontajärjestelmää. Systeemirajan sisällä oleva toiminnallisuus on sellaista, joka voidaan realisoida käyttäytymistä kuvaavilla kaavioilla kuten aktiviteettikaaviolla, sekvenssikaavioilla tai tilakonekaavioilla. [50]

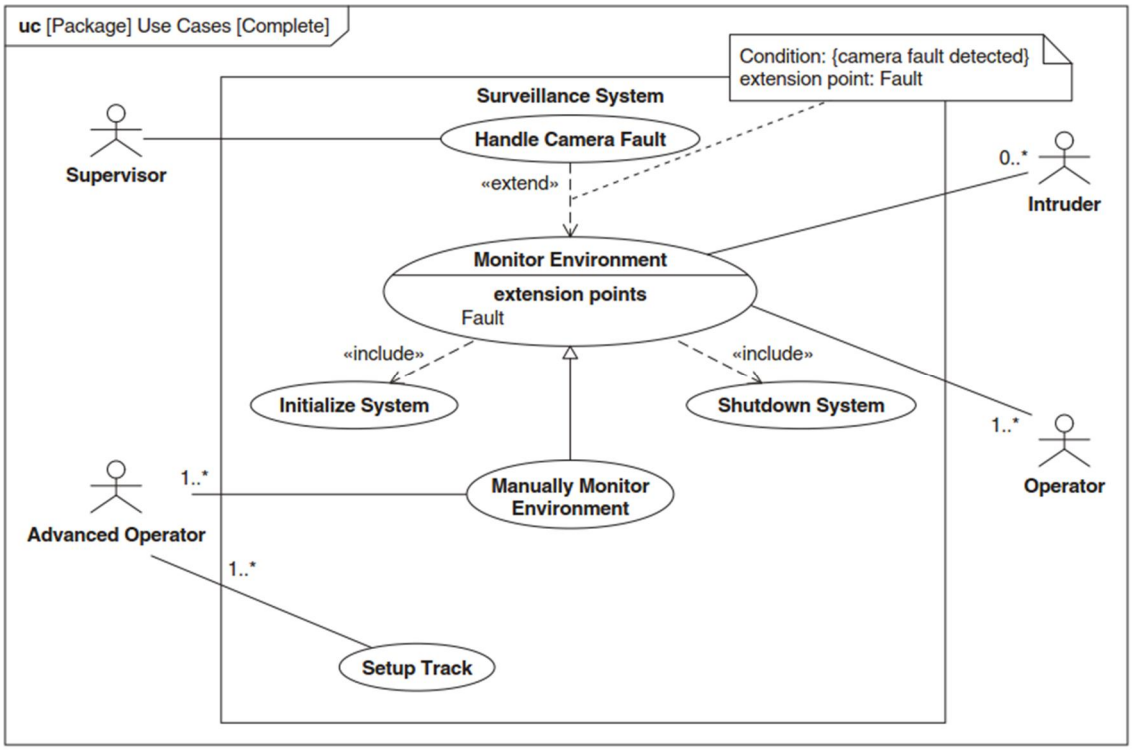
Mallielementtien välisiä suhteita käyttötapauskaaviossa voidaan kuvata neljällä erilaisella tavalla. Kommunikointipolku ilmaistaan assosiaatioilla. Käyttötapauskaaviossa

tämä kuvataan suoralla yhtenäisellä viivalla toimijan ja käyttötapauksen välissä. Toimijan tai käyttötapauksen päässä voidaan ilmaista kerrannaisuus.

Yleistämisellä voidaan käyttötapauskaavioissa määritellä erikoistapauksia peruskäyttötapauksesta. Yleistäminen ilmaistaan kaaviossa yhtenäisellä nuolella yleistettävän mallielementin suuntaan. [50] Yleistämistä voidaan käyttää myös toimijoiden kohdalla. Kuvassa 19 on esitetty yleistetty manuaalista ympäristön valvontaa kuvaava käyttötapaus normaalista valvontaa kuvaavasta käyttötapauksesta.

Sisältymissuhde (*included use case*) voidaan ilmaista käyttötapauskaaviossa <<include>> avainsanalla ja katkoviivalla, jonka päässä on avoin nuolenkärki. Nuolenkärki osoittaa sisältyvän käyttötapauksen suuntaan. Sisältyvä käyttötapaus suoritetaan, jos sen lähteenä oleva käyttötapaus suoritetaan. Sisältyvä käyttötapaus on siis vaadittu osa pääkäyttötapausta. [18] Sisältymissuhteesta käyttötapauskaaviossa on esimerkki kuvassa 19.

Laajennussuhde (*extended use case*) kuvataan käyttötapauskaaviossa katkoviivalla ja avoimella nuolenkärjellä pääkäyttötapauksen suuntaan. Lisäksi käytetään avainsanaa <<extend>>. Laajennettu käyttötapaus suoritetaan vain tarvittaessa. Pääkäyttötapaukseen voidaan määritellä tapahtumat (*extension points*), jotka aiheuttavat mahdolliset laajennetut käyttötapaukset. [18] Edellä mainittu menetelmä käyttötapauskaaviossa on esitelty kuvassa 19.



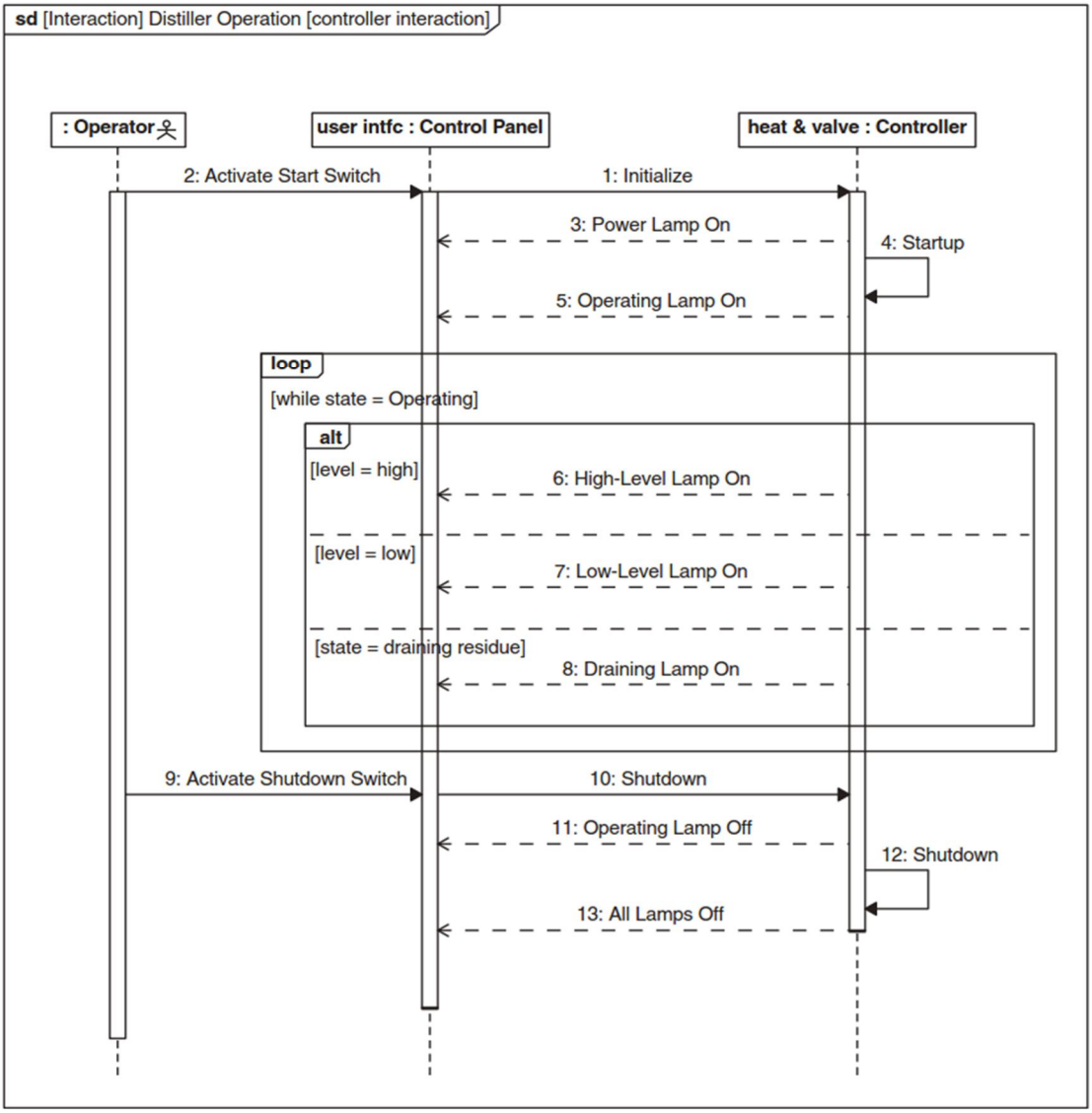
Kuva 19. *Käyttötapauskaavio [40].*

Käyttötapauskuvausten laatimiseen ei ole UML- tai SysML-kielissä annettu mitään tarkkaa määrittelyä, vaikka ne ovat tärkeä osa käyttötapauksen havainnollistamaa toiminnallisuutta [41]. Määrittelyjen muodostaminen on hyvin pitkälle kiinni mallintajan tavoista ja yhteisistä sopimuksista. Kirjallisuudesta löytyy kuitenkin ohjeistuksia erilaisten käyttötapausten laatimiseen [18,41,51].

Käyttötapauskaavioiden tukena tekstipohjaisten käyttötapauskuvausten sijaan tai niiden lisäksi voidaan käyttää SysML-kielen käyttäytymistä kuvaavia kaavioita. Sekvenssikaavio soveltuu käyttötapauksen kuvaamiseen, kun sen sisältämä skenaario on suurilta osin viestipohjainen. Aktiviteettikaavio on käytännöllinen silloin, kun skenaarioon liittyy kontrollilogiikkaa, vuopohjaisuutta tai dataa muokkaavia algoritmeja. Tilakonekaaviota voidaan käyttää silloin, kun toimijoiden ja systeemin välinen vuorovaikutus on asynkronista, eikä sitä ole helppoa esittää tiettyjen tapahtumien järjestyksenä. [40]

3.8.5 Sekvenssikaavio

Sekvenssikaavion tarkoituksena on kuvata vuorovaikutusta systeemin eri rakenne-elementtien välillä [40]. Kaavio voi kuvata ainoastaan vuorovaikutus-mallielementtiä. Sekvenssikaavio on käytännöllinen tapa kuvata systeemin käyttäytymistä varsinkin mallinnettaessa palvelukeskeistä toimintaa. Kaaviossa systeemin rakenne-elementti kutsuu toisen elementin tarjoamia palveluja viestipohjaisesti. [40 s.251] Tyypillisesti sekvenssikaavio kuvaa systeemin yksittäisen skenaarion [41]. Esimerkkitapaus sekvenssikaaviosta on kuvassa 20.



Kuva 20. *Sekvenssikaavio [40].*

Elämänviiva (*lifeline*) on elementti, joka kuvaa osanottajaa vuorovaikutuksessa. Elämänviivat vastaavat vuorovaikutuksen omistavan lohkon osa-ominaisuuksia. [18] Elämänviivaa kuvataan sekvenssikaaviossa nelikulmiolla, josta lähtee katkoviiva. Tämä katkoviiva kuvaa osa-ominaisuuden elinaikaa. Elämänviivalla voi tapahtua kuusi erilaista tapahtumaa:

- Viestin lähetys
- Viestin vastaanotto
- Elämänviivan luominen
- Elämänviivan tuhoaminen
- Käyttäytymisen suorituksen aloittaminen
- Käyttäytymisen suorituksen lopettaminen

Sekvenssikaaviossa elämänviivojen välinen vuorovaikutus tapahtuu viesteillä. Viestit voivat olla asynkronisia tai synkronisia. Asynkroninen viesti tarkoittaa, että viestiä lähettävän elementin ei tarvitse odottaa vastausta jatkaakseen toimintansa suorittamista. Synkronisen viestin lähettänyt elementti odottaa vastauksen ennen kuin jatkaa toimintaansa. [41] Sekvenssikaaviossa asynkroninen viesti kuvataan avoimella nuolenkärjellä. Synkronista viestiä mallinnettaessa nuolenkärki on suljettu. Vastausviesti on katkoviivalla kuvattu nuoli avoimella kärjellä. [40 s.257] Elementin elinajan loppuminen sekvenssikaaviossa ilmaistaan isolla rastilla.

Yleensä viestit sekvenssikaaviossa mallintavat informaatiota, joka siirtyy ohjelmistojen ja niiden käyttäjien välillä. Viesteillä voidaan kuitenkin mallintaa esimerkiksi materiaalin tai energian siirtymistä systeemissä. [40 s.256]

Alisekvenssit (*combined fragment*) ovat kuvaustekniikka, jolla voidaan lisätä ohjauslogiikkaa sekvenssikaavioihin. Tällä tavoin voidaan mallintaa esimerkiksi vaihtoehtoista suoritusta, silmukoita tai rinnakkaista suoritusta. Alisekvenssi mallinnetaan sekvenssikaaviossa suorakulmiolla, jonka sisällä suoritettavan alisekvenssin tapahtumat ovat. Suorakulmion vasemmassa yläkulmassa sijaitseva operaattori selittää, minkälaista logiikkaa alisekvenssi suorittaa. SysML-kielessä määritellään 11 operaattoria. Näistä operaattoreista neljä on kuitenkin useimmiten käytössä vuoropohjaista toiminnallisuutta mallinnettaessa. [18]

Opt operaattorin avulla voidaan mallintaa vaihtoehtoisia tapahtumia alisekvenssin sisällä.

Alt operaattoria käyttämällä voidaan määritellä kaksi tai useampia vaihtoehtoisia tapahtumakokonaisuuksia, jotka voivat tapahtua vuorovaikutuksen aikana. Vaihtoehtoisen suorituksen alisekvenssisestä on esimerkkitapaus kuvassa 20.

Loop operaattorilla voidaan mallintaa suoritusta, jota niin kauan kuin annetut ehdot ovat voimassa. Tästä tapauksesta on esimerkki kuvassa 20.

Par operaattorin avulla voidaan mallintaa rinnakkaista toimintaa alisekvenssissä. Kyseinen alisekvenssi sisältää kaksi tai useamman tapahtuman, joka suoritetaan rinnakkain. Normaalisti sekvenssikaaviossa ylempi tapahtuma suoritetaan aina ensin. Par operaattorilla merkityssä alisekvenssissä tapahtumat voivat tapahtua missä tahansa järjestyksessä.

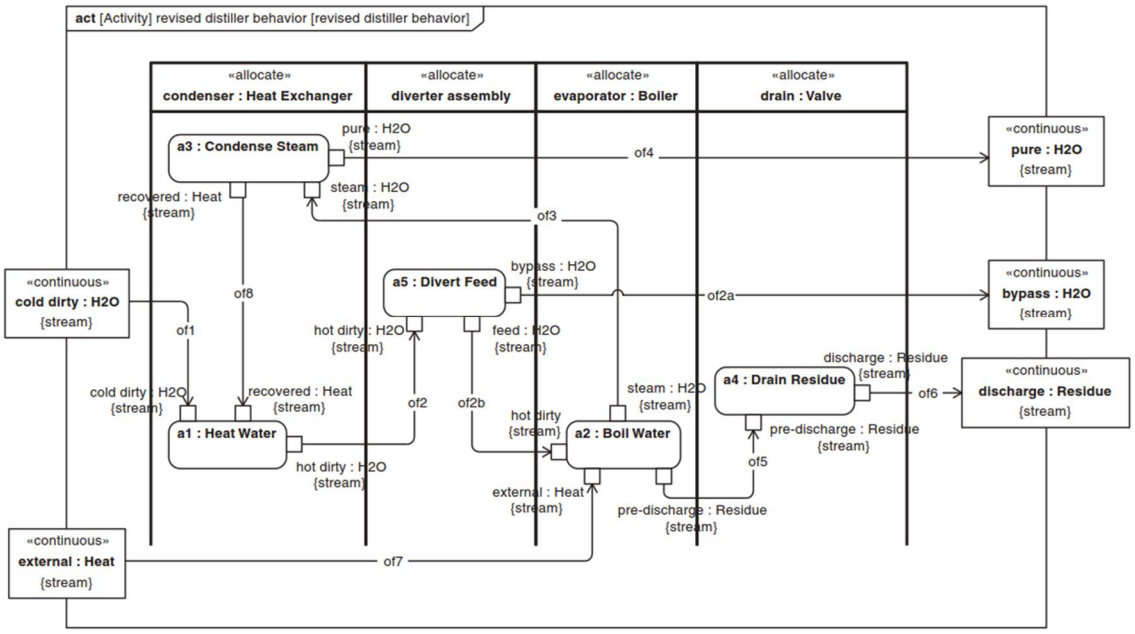
3.8.6 Aktiviteettikaavio

Aktiviteettikaavio on yksi kolmesta SysML kaaviosta, joiden avulla voidaan mallintaa systeemin dynaamista käyttäytymistä. Aktiviteettikaavion lyhenne on **act**. Ainoa sallittu mallielementti, jota kaavio voi kuvata, on aktiviteetti. Aktiviteetit koostuvat toiminnois-

ta (*action*). Toiminnot kuvaavat aktiviteettien suoritusta ja kuinka sisääntulot muuttuvat ulostuloiksi. [40]

SysML-kielen aktiviteettikaavio on laajennettu versio UML-kielen aktiviteettikaaviosta. Kaaviossa voidaan mallintaa vuopohjaista käyttäytymistä. Vuopohjaista toimintaa mallinnetaan virtaavilla tunnuksilla (*tokens*). Tunnuksia on kahta erilaista tyyppiä. Oliotunnus (*object token*) esittää materiaa, energiaa tai dataa, joka virtaa aktiviteetin läpi. Ohjaustunnus (*control token*) määrittää, mikä toiminto aktiviteetissa suoritetaan. Oliovuota kuvataan aktiviteettikaaviossa kiinteällä viivalla, jonka nuolenkärki osoittaa vuon suunnan. Ohjausvuota kuvataan katkoviivalla, jossa siinäkin voidaan osoittaa suuntaa nuolella. [18]

Aktiviteettikaavion esittämät aktiviteetit voidaan jaotella systeemin muille osille esimerkiksi lohkoille tai toimijoille aktiviteetin osituksen avulla (*activity partition*). Aktiviteetin osituksella pystytään osoittamaan aktiviteetin eri toiminnallisuuksia systeemin rakenteelle. Käytännössä ositus osoitetaan aina jollekin systeemin lohkolle tai toimijalle, kuten on tehty kuvan 21 esimerkissä, joka mallintaa veden tislausprosessia.

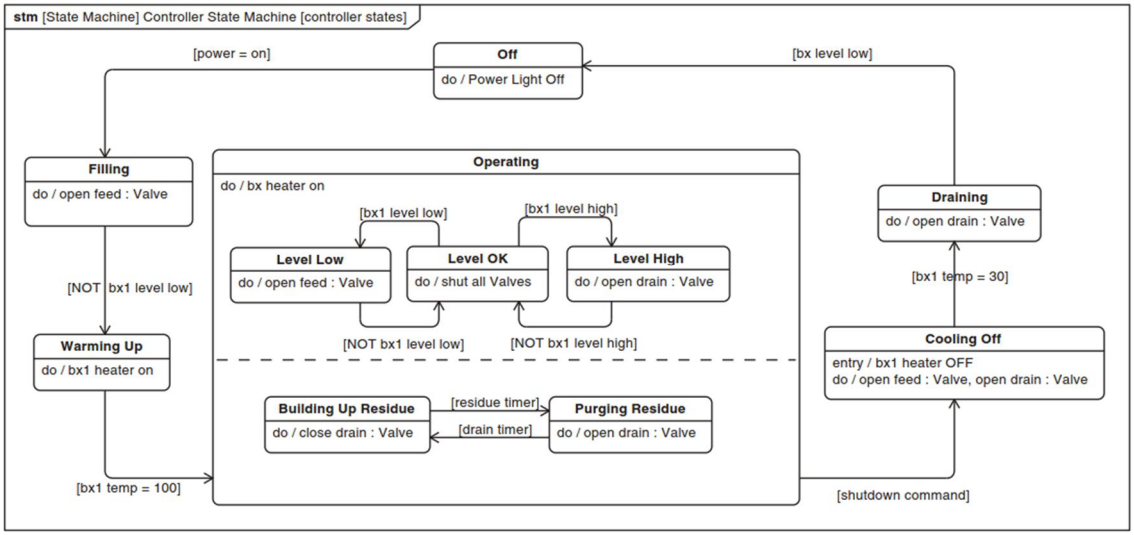


Kuva 21. *Aktiviteettikaavio[40].*

Aktiviteettikaaviota voidaan käyttää myös käyttötapausten sisältämien skenaarioiden graafiseen mallintamiseen. Tämä on kuitenkin usein sopimuksellinen asia mallinnusmenetelmästä riippuen. Tällöin käyttötapauksista systeemissä muodostetaan erilliset aktiviteettikaaviot. [18] Myös testitapauskuvausten määrittäminen aktiviteettikaavioiden avulla on mahdollista.

3.8.7 Tilakonekaavio

Tilakonekaavio on viimeinen kolmesta SysML-kielen kaaviosta, joilla voidaan mallintaa systeemin dynaamista käyttäytymistä. Systeemillä tai sen osalla on tietty määrä erilaisia tiloja, joissa se voi olla systeemin toiminnan aikana. Tilakonekaavio kuvaa systeemin rakenteen tilojen välisiä muutoksia. Yleensä tilakonekaaviolla mallinnetaan jonkin lohkon käyttäytymistä. Kaavio kuvaa usein kohteen koko eliniän tilakäyttäytymistä. Tilakone on itsessään mallielementti. Se on myös nimiavaruus ja voi sisältää muita nimettyjä elementtejä. Esimerkki tilakonekaaviosta on esitetty kuvassa 22.



Kuva 22. *Esimerkki tilakonekaaviosta[40].*

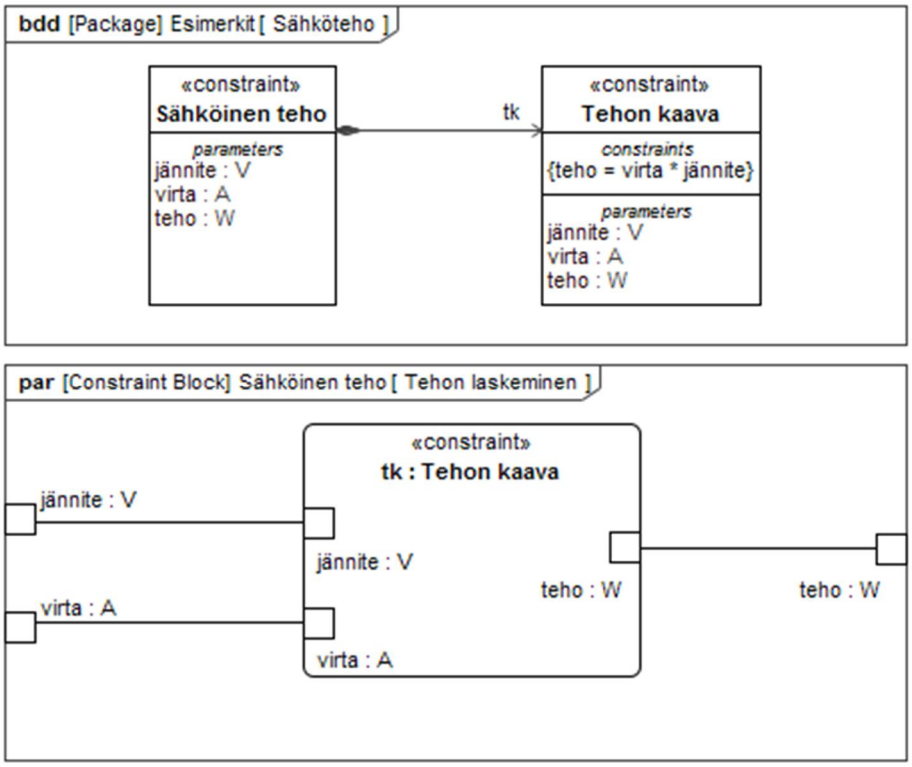
Jokaiselle tilalle voidaan SysML-kielessä määrittää kolme erilaista sisäistä käyttäytymistä: entry, exit ja do. Entry suoritetaan silloin, kun tilaan siirrytään. Exit silloin, kun tilasta siirrytään pois. Tilalle voi olla määritetty myös pelkästään do käyttäytyminen, mikä suoritetaan kerran, kun tilaan siirrytään. Suoritusta jatketaan, kunnes se valmistuu tai tilasta poistutaan.

Tilakonekaavioilla kuvataan systeemin erillisen osan, alisysteemin tai koko systeemin toimintaa. Siksi tilakonekaavioita voidaan luoda missä tahansa systeemin elinkaaren vaiheessa selventämään systeemin suunnittelua. [18]

3.8.8 Parametrinen kaavio

Lohkon määrittelykaavion yhteydessä esitelty rajoitelohko on erityinen lohko, joka määrittää jonkin mallinnuksessa tarvittavan yhtälön. Parametrinen kaavio on SysML-kielessä määritelty erityiseksi versioksi sisäisestä lohkokaaviosta. Parametrinen kaavio voi esittää mallielementtinä lohkoa tai rajoitelohkoa. Kaavion avulla kuvataan yleensä

yhteyksiä lohkon arvo-ominaisuuksien tai rajoiteparametrien välillä. Parametrisen kaavion käyttö vaatii yleensä tarkkoja yksityiskohtia systeemin mallin suhteen, eikä sitä välttämättä tarvitse käyttää jokaisen mallinnusprojektin yhteydessä. [18] Kuvassa 23 on yksinkertaisessa esimerkissä esitetty sähköisen tehon kaavaa kuvaava rajoitelohko sekä sitä mallintava parametrinen kaavio.



Kuva 23. *Rajoitelohko ja sitä kuvaava parametrinen kaavio.*

3.8.9 Vaatimuskaavio

Vaatimuskaavio SysML-kielessä esittää tekstipohjaiset vaatimukset sekä niiden suhteet muihin vaatimuksiin, suunnitteluelementteihin ja testitapauksiin. Kaavio on lisätty aiempiin UML-kirjastoihin juuri systeemisuunnittelun tarpeita ajatellen. Vaatimuskaavio on tarkoitettu varsinkin systeemin ei-toiminnallisten vaatimusten kuvaamiseen. Mallielementit, joita vaatimuskaavio voi esittää: ovat pakkaus, malli, mallikirjasto, näkymä tai vaatimus. Varsinkin toiminnallisten vaatimusten mallintaminen käyttötapausten avulla on suosittua, mutta tarpeeseen tekstipohjaisille vaatimuksille vaatimuskaavio on hyvä työkalu [18 s.201]. Vaatimuskaavio on käytännöllinen varsinkin silloin, kun on tarve osoittaa vaatimusten jäljitettävyys niihin liittyviin systeemin mallin elementteihin [18 s.202].

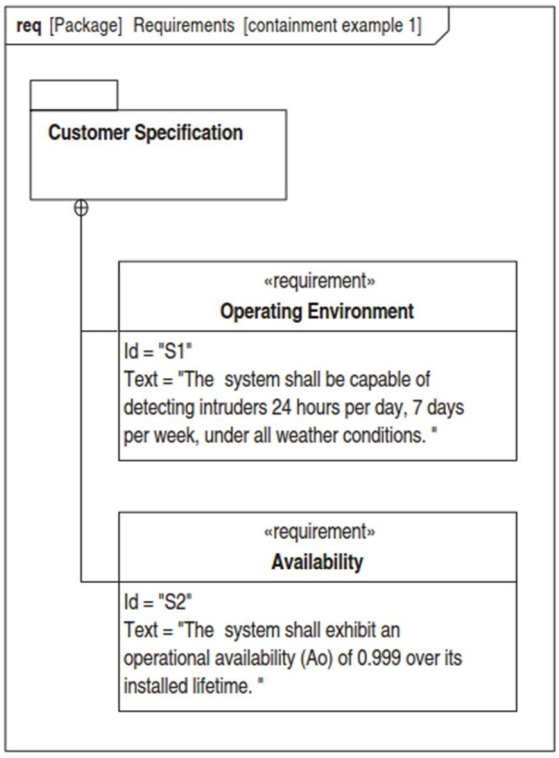
Ominaisuuksia vaatimuksella on kaksi: id ja text. Vaatimus tunnistetaan id:n perusteella ja text-kentässä kuvataan itse vaatimus. Molemmat ominaisuudet ovat merkkijono-

tyyppiä. Merkkijonojen sisällöt ovat täysin mallintajan päätöksessä. Tunnisteiden numerointi tai nimeäminen on kuitenkin syytä tehdä johdonmukaisesti, jotta sen perusteella voidaan tunnistaa toisiinsa liittyvät vaatimukset.

SysML kielessä ei ole määritetty kuinka vaatimukset tulisi mallintaa. Mallintaja voi tehdä yhden erillisen vaatimuksen, jossa on koko vaatimusmäärittely. Toinen käytetty tapa on muodostaa pakkaus, jossa vaatimukset on eritelty. Monet työkalut tekevät tämän oletuksena vaatimuksia malliin tuotaessa. Olisi kuitenkin hyvä, että tietyn organisaation sisällä sovitaan yhtenäiset tavat, kuinka vaatimuksia mallinnetaan tekstipohjaisesti. [18]

Vaatimusten väliset suhteet muihin mallielementteihin ovat tärkeimpiä seikkoja, jotka täytyy ottaa huomioon vaatimuksia mallinnettaessa. Yleisimmin käytetyt vaatimusten väliset suhdetyypit ovat: **containment**, **trace**, **derive requirement**, **refine**, **satisfy** ja **verify**. Nämä suhteet muodostavat jäljitettävyyden vaatimuksille. Lisäksi SysML-kielessä on määritelty **copy**-suhde, mutta se on harvemmin käytetty. [18]

Containment eli sisältyvyyssuhde voidaan esittää samalla tavoin kuin pakkausten tapauksessa tähtäinnotaatiolla kuten kuvassa 24 on esitetty tai tarkalla nimiavaruuden nimellä. Vaatimus voi kuitenkin sisältää ainoastaan muita vaatimuksia.



Kuva 24. *Vaatimusten sisältyvyys [40].*

Vaatimusten välisiä riippuvuussuhteita voidaan ilmaista kuvan 25 tavalla tai lokeronotaatiolla, jossa vaatimuselementissä on selitetty mihin toiseen elementtiin vaatimuksella on riippuvuussuhde.

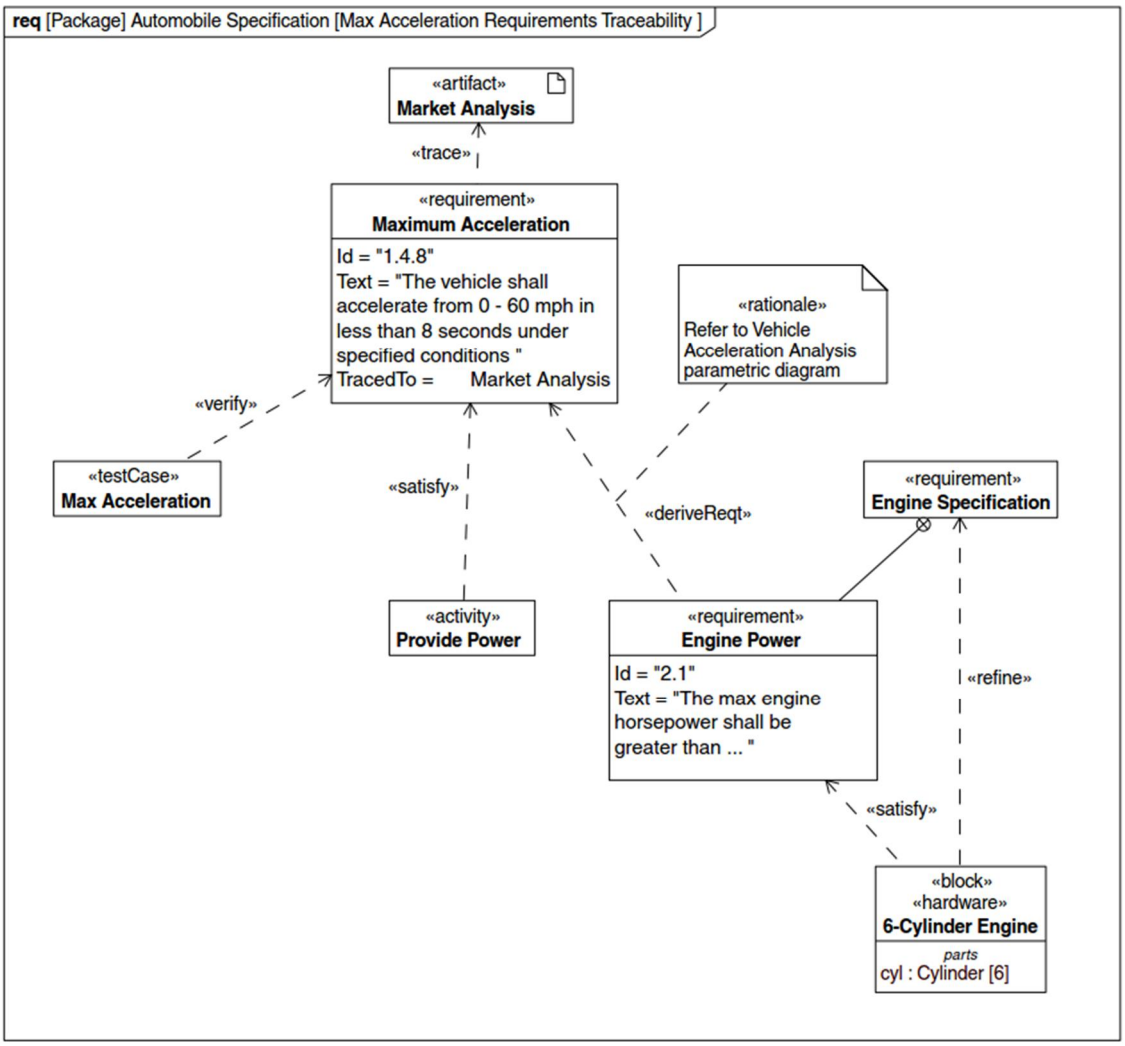
Trace riippuvuussuhde ilmaisee vaatimuksen jäljitettävyyden johonkin mallielementtiin. Kuvassa 25 kiihtyvyyden vaatimuksella on kyseinen riippuvuussuhde markkina-analyysiin.

Derive requirement riippuvuussuhteella ilmaistaan, että vaatimus on johdettu jonkin toisen vaatimuksen pohjalta. Riippuvuus on kuvattu kaaviossa avainsanalla <<deriveReqt>>. Johdetun riippuvuuden ei tarvitse olla samassa nimiavaruudessa kuin sen tarjoajan. Tämän riippuvuussuhteen täytyy mallissa olla aina kahden vaatimuksen välinen. [18] Kuvan 25 vaatimuskaaviossa on johdettu moottorin tehovaatimus kiihtyvyyden vaatimuksen pohjalta.

Refine riippuvuussuhteella ilmaistaan suhdetta johonkin toiseen mallielementtiin, jonka avulla voidaan selventää vaatimusta.

Satisfy riippuvuussuhteella esitetään, että mallielementti täyttää vaatimuksessa esitetyt asiat. SysML-kielessä ei eritellä, minkälainen mallielementin täytyy olla, mutta sen täytyy toteuttaa vaatimus. Satisfy-riippuvuussuhde on ainoastaan tapa osoittaa vaatimus rakenne-elementille. Vaatimuksen lopullinen toteutuminen varmistetaan testitapauksen avulla. [18]

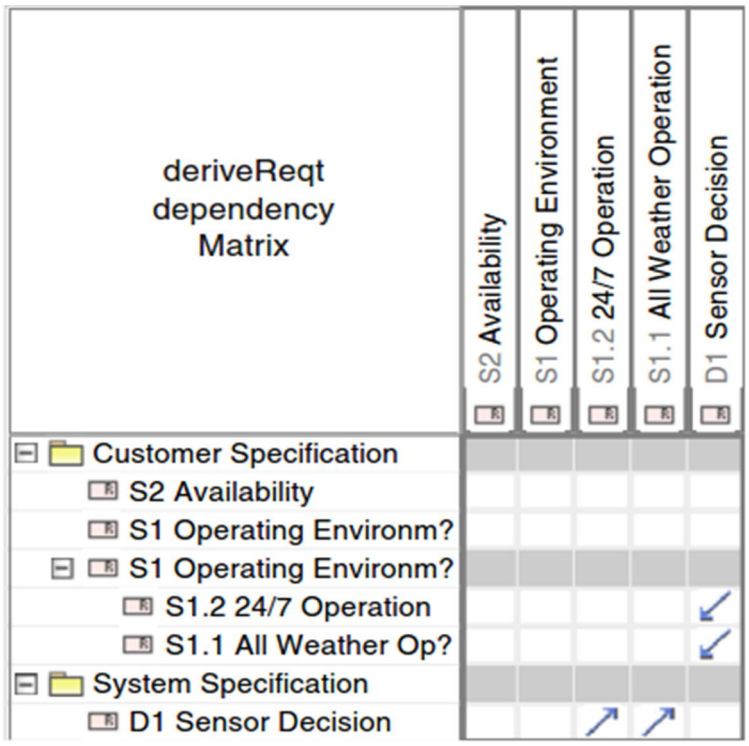
Verify riippuvuussuhteella voidaan ilmaista, kuinka mallielementti varmistaa vaatimuksen toteutumisen. SysML-kielessä ei ole rajoituksia, mikä toteuttavan mallielementin täytyisi olla, mutta useimmiten se on testitapaus. Testitapaus on SysML-kielessä jokin käyttäytymistä kuvaava mallielementti eli aktiviteetti, vuorovaikutus tai tilakone. Useimmiten testitapaus on kuitenkin mallinnettu vuorovaikutuksena ja esitetty sekvenssikaaviossa. [18]



Kuva 25. *Vaatimusten väliset suhteet vaatimuskaaviossa [40].*

Vaatimusten määrittelyssä voidaan käyttää hyväksi perustelua (*rationale*), kuten on tehty kuvassa 25. Perustelu on SysML-kielessä erityinen kommentti. Mallielementin graafinen esitys on kommentin tapaan muistiinpanolappu, jonka oikea yläkulma on taitettu ja lapussa on <<rationale>> avainsana. Kommentin tai perustelun voi liittää myös mihin tahansa muuhun mallielementtiin SysML-kielen eri kaavioissa.

SysML-kielessä voidaan käyttää matriisiesitystä vaatimusten sekä muiden suunnitteluelementtien välisiä suhteita ilmaistaessa. Matriisiesityksen huonona puolena on, ettei siinä näy elementtien piirteitä. Se ilmaisee ainoastaan niiden väliset suhteet. [18] Kuvassa 26 on esimerkki matriisiesityksestä.



Kuva 26. *Vaatimusten suhteiden matriisiesitys [40].*

Taulukkoesitys on matriisin kaltainen esitystapa vaatimuksille, josta on esimerkki kuvassa 27. Taulukoissa voidaan esittää pelkästään listauksia vaatimuksista tai mallintaa niiden välisiä suhteita. Monissa työkaluissa käyttäjä voi itse luoda haluamansa suuruisen ja tyyppisen taulukon. [18 s.213]

table [Package] System Specification [Decomposition of top-level requirements]

id	name	text
S1	Operating Environment	The system shall be capable of detecting intruders 24 hours per day…
S1.1	All Weather Operation	The system shall be capable of detecting intruders under all weather…
S1.2	24/7 Operation	The system shall detect intruders 24 hours per day, 7 days per week
S2	Availability	The system shall exhibit an operational availability (Ao) of 0.999…

Kuva 27. *Vaatimusten taulukkoesitys [40].*

3.9 Mallipohjaisen systeemisuunnittelun menetelmät

Systeemisuunnittelun prosessit määrittävät mitä kussakin systeemin elinkaaren vaiheessa täytyy saavuttaa. Menetelmä kertoo, kuinka se saavutetaan. Mallinnusmenetelmään liittyy toimintoja, tekniikoita ja yleisiä käytäntöjä, joita käytetään suunnittelun edetessä. Menetelmän valinta riippuu siitä, minkä tyyppistä systeemiä ollaan suunnittelemassa ja mihin tarkoitukseen mallia pääasiassa käytetään. Suunnittelutahon täytyy arvioida itse mitä heidän suunniteltavan systeemin malli vaatii ja mikä on sen käyttöalue. Näihin

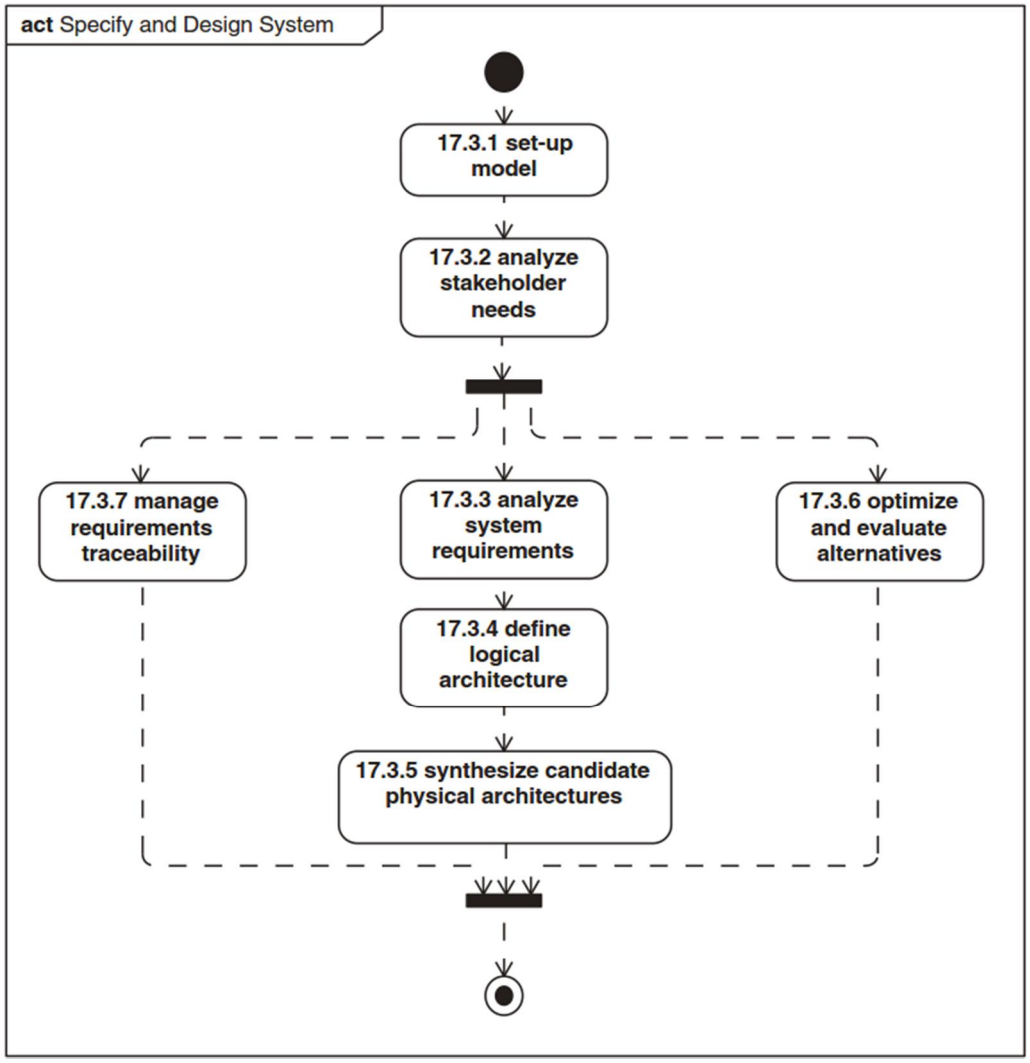
vastaamalla voidaan valita käytettävä menetelmä. Muutamia käytettyjä mallipohjaisen systeemisuunnittelun menetelmiä ovat:

- INCOSE Object-Oriented Systems Engineering Method (OOSEM) [40]
- Systems Modeling Process (SYSMOD) [23]
- IBM Rational Unified Process for Systems Engineering (RUP-SE) [42]

Eri menetelmät ovat mallinnusvaiheidensa osalta samankaltaisia ja sisältävät useimmat systeemisuunnittelun elinkaariprosesseista. Yleensä ne toimivat hyvänä pohjana projektin mallinnusta suunniteltaessa. Menetelmä joudutaan kuitenkin melkein aina muokkaamaan projekti- tai organisaatiokohtaiseksi ja juuri omaan tarkoitukseen sopivaksi [18,43].

3.9.1 OOSEM

OOSEM on ISO/IEC 15288 systeemisuunnittelun standardiin pohjautuva menetelmä. Standardi määrittää prosessit, jotka kertovat mitä suunnittelun pitää tuottaa. Menetelmä määrittää kuinka se tehdään. OOSEM on skenaariopohjainen menetelmä systeemien analysointiin, määrittelyyn, suunnitteluun ja verifiointiin. Menetelmää käytetään pääasiassa luvussa 2.5 esitellyissä ISO/IEC 15288 standardin mukaisissa systeemin elinkaaren konsepti- ja kehitysvaiheissa. OOSEM-menetelmän systeemin suunnittelu- ja määritysvaiheeseen kuuluvat päävaiheet ovat esitelty kuvassa 28 näkyvässä aktiviteettikaaviossa.



Kuva 28. *OOSEM, systeemin määrittely ja suunnittelu [40].*

Ensimmäiseksi täytyy perustaa malli. Tämän vaiheen aikana päätetään mallinnustavoista ja yhteisistä säännöistä, joita mallin muodostamisessa käytetään. Tässä vaiheessa päätetään myös, kuinka systeemin malli organisoidaan mallihierarkiassa. Organisointiin käytetään erilaisia pakkauksia. OOSEM määrittelee oletusjärjestyksen mallin rakenteeksi, jota mallintava taho voi halutessaan käyttää. Tämä rakenne tukee OOSEM-menetelmän eri vaiheita ja on siksi helppokäyttöinen ratkaisu. [40]

Ensimmäisenä varsinaisena suunnitteluvaiheena OOSEM-mentelmässä on sidosryhmien tarpeiden analysointi. Tässä vaiheessa selvitetään systeemin sidosryhmät ja määritetään haluttu toiminnallisuus sidosryhmien tarpeiden perusteella. Muodostetaan yksinkertainen toimialamalli, jotta voidaan tunnistaa suunniteltava systeemi, ulkoiset systeemit ja käyttäjät. Toiminnallisuutta mallinnetaan korkean tason käyttötapauksilla. Toiminnallisuuden onnistumiselle määritetään luvussa 3.3.1 esiteltyjä vaikuttavuuden mittarit. Tässä vaiheessa karakterisoidaan myös mahdollinen olemassa oleva systeemi ja määritetään sen perusteella systeemin puutteet ja kehityskohteet. Karakterisoinnin avulla saadaan

selville myös mahdolliset uudelleen käytettävät systeemin osat. Samalla voidaan selvittää myös muiden samankaltaisten järjestelmien ominaisuuksia. [40]

Sidosryhmätarpeiden pohjalta siirrytään OOSEM-mentelmän seuraavaan vaiheeseen, joka on systeemivaatimusten analyysi. Kyseisessä vaiheessa systeemi ajatellaan black box-mallina, jossa määritellään systeemille sisään- ja ulostulot eli mitä systeemin pitää tuottaa. Lisäksi määritellään systeemin ulkoiset rajapinnat. Systeemin suorituskykyä mittaavat mittarit johdetaan tässä vaiheessa sidosryhmätarpeissa määriteltyjen vaikuttavuuden mittareiden pohjalta. [40]

Loogisen arkkitehtuurin määrityksessä systeemin sisäinen toiminta jaotellaan loogisiksi komponenteiksi, jotka toteuttavat systeemivaatimukset. Loogiset osat ovat fyysisten komponenttien abstraktioita, jotka eivät vielä sisällä rajoituksia toteutuksesta. Lisäksi määritellään loogisten komponenttien väliset vuorovaikutussuhteet. Varsinaista fyysistä ratkaisua ei tässä vaiheessa vielä tiedetä.

Viimeisessä vaiheessa systeemin toiminnalle määritetään fyysinen ratkaisu. Ensin jo määritellyt loogiset osat ryhmitellään kokonaisuuksiin niiden sijoituspaikan mukaan. Tämän jälkeen loogisten komponenttien toiminta jaotellaan fyysisille komponenteille. Fyysisille osille määritetään osavaatimukset. Toteutuksen kannalta kriittisille komponenteille voidaan määrittää teknisen suorituskyvyn mittareita suorituskyvyn mittareiden perusteella. Systeemin toteuttavat osat valitaan osavaatimusten ja teknisten suorituskykymittareiden perusteella.

Kolmen viimeisen vaiheen rinnalla suoritetaan optimointia eri toteutusvaihtoehdoista. Kriteereinä eri toteutusten välillä voidaan käyttää luvussa 3.3.1 esiteltyjä teknisen suorituskyvyn mittareita. Erilaisia toteutuksia voidaan vertailla esimerkiksi SysML-kielen parametrisen kaavion avulla. [21,40]

Lisäksi vaatimusten jäljitettävyyden hallinta toteutetaan kaikkien menetelmän vaiheiden aikana sidosryhmävaatimuksista lähtien lopulliseen systeemin arkkitehtuuriin asti. Jäljitettävyyttä voidaan kuvata luvussa 3.7.8 esitellyn SysML-vaatimuskaavion ja sen riippuvuussuhteiden avulla.

OOSEM toimii myös systeemin valmistuneen toteutuksen ja erilaisten osakomponenttien integroinnin ja verifioinnin tukena. Menetelmän avulla muodostettua systeemin mallia voidaan käyttää pohjana luotaessa testitapauksia ja verifiointiprosesseja. Systeemin malli toimii myös yhdistävänä tekijänä ja pohjana muille mallinnusmenetelmille, jotka tukevat verifiointia, kuten simuloitavat analyyttiset mallit. Myös systeemin elinkaaren myöhempiä vaiheita kuten asennusta voidaan tukea systeemin mallin avulla. [40]

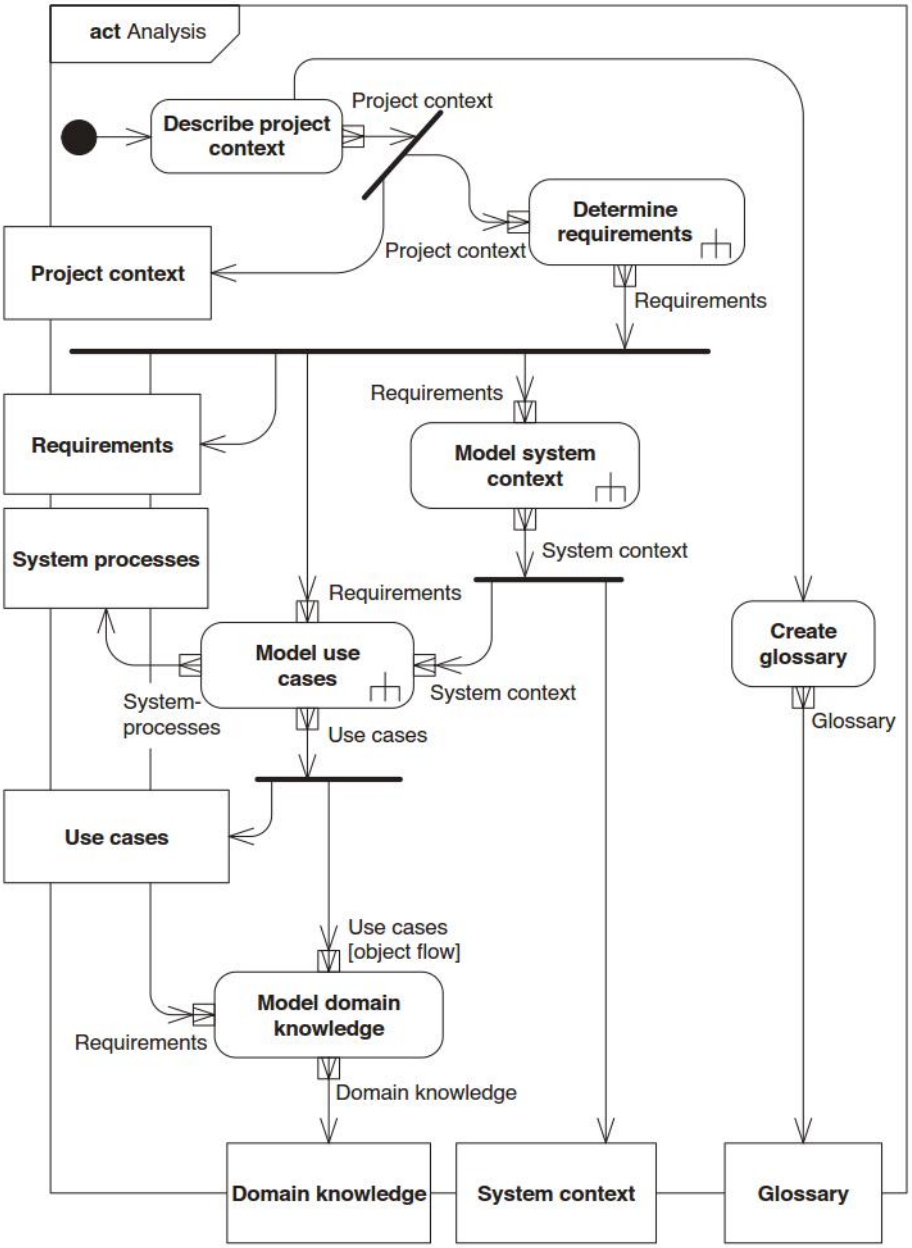
Menetelmän ohella on olemassa erillinen OOSEM SysML-profiili, joka sisältää systeemin mallin suunnittelussa käytettäviä stereotyyppejä. Profiili määrittelee esimerkiksi

teknisiä mittareita, joita voi käyttää päättäessä erilaisista ratkaisuvaihtoehdoista. Suunnittelu onnistuu kuitenkin pelkästään menetelmän perusohjeita noudattamalla ja käyttämällä SysML-peruskirjastoja.

3.9.2 SYSMOD

SYSMOD on Weilkinsin kirjassaan [23] esittelemä SysML-kieleen pohjautuva mallipohjaisen systeemisuunnittelun menetelmä. SYSMOD perustuu vahvasti käyttötapausten perusteella tehtävään systeemin rakenteen suunnitteluun. Menetelmään on erillinen SYSMOD-profiili, joka sisältää menetelmässä käytettäviä stereotyyppejä SysML-kielen vakioelementtien tueksi.

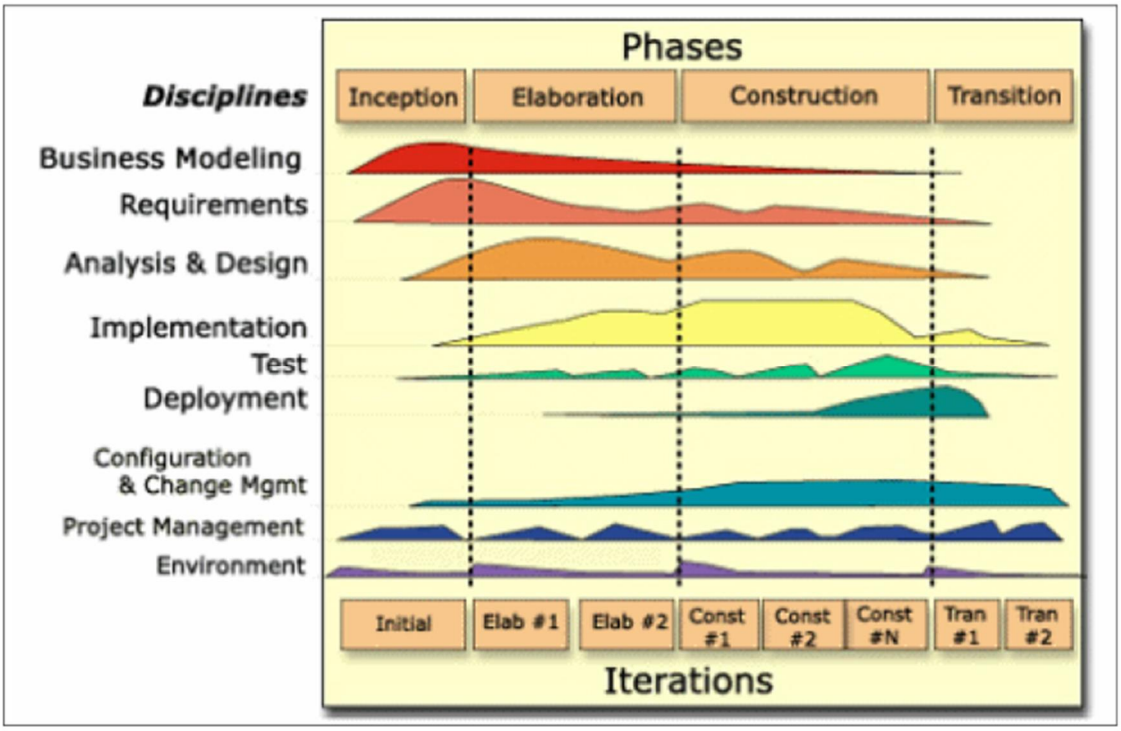
Menetelmän perusvaiheet ovat esitelty kuvassa 29. Menetelmässä määritellään ensin projektin sisältö, jonka perusteella selvittää vaatimukset ja projektiin liittyvä sanasto. Vaatimusten perusteella voidaan mallintaa systeemin sisältöä sekä käyttötapauksia. SYSMOD on menetelmänä hyvin samankaltainen kuin edellisessä alaluvussa esitelty OOSEM-menetelmä.



Kuva 29. *SYSMOD-menetelmä [23].*

3.9.3 RUP-SE

RUP-SE on IBM:n markkinoima projektikohtaisesti muokattava kehys mallipohjaiseen systeemisuunnitteluun. Se perustuu RUP-menetelmään (*Rational Unified Process*), joka on yleisesti käytetty suunnittelukehys ohjelmistosuunnittelussa. RUP elinkaari on luvussa 2.6.2 esitellyn ohjelmistotuotannon iteratiivisen spiraalimallin kaltainen [76]. RUP-menetelmän elinkaari sisältää neljä vaihetta: aloitus (*Inception*), tarkennus (*Elaboration*), rakennus (*Construction*) ja käyttöönotto (*Transition*) [30]. Prosessikehikko sisältää yhdeksän työkokonaisuutta, joita suoritetaan elinkaaren aikana. Eri työkokonaisuudet prosessin elinkaaren vaiheissa ovat esitetty kuvassa 30.



Kuva 30. *RUP prosessikehikko [42]*

Pääelementit RUP:in sisällössä ovat roolit (*Roles*), tuotteet (*Work Products*) ja tehtävät (*Tasks*). Roolit määrittävät prosessissa olevien henkilöiden tarvittavat taidot, pätevyydet ja vastuualueet. Tuotteet esittävät tuloksia, joita prosessin eri vaiheissa täytyy tuottaa, kuten erilaisia dokumentteja ja malleja. Tehtävät määrittävät, kuinka työ jaetaan eri roolien kesken, että päästään haluttuun lopputulokseen. [76]

RUP-SE laajentaa normaaliin ohjelmistosuunnitteluun tarkoitetun RUP-menetelmän sisältöä systeemisuunnittelun tarpeisiin. Muun muassa suunnittelurooleihin sisältyy systeemisuunnittelijoita normaalien kehittäjien ja testaajien sijaan. RUP-SE arkkitehtuurikehikko määrittää myös erilaisia suunnittelutuotteita ja työnkulkuja, jotka sopivat paremmin systeemisuunnittelun pariin. [76]

3.10 MBSE systeemin uudelleensuunnittelussa

Mallipohjaista systeemisuunnittelua käytetään usein monimutkaisten järjestelmien suunnitteluun. MBSE-menetelmät tarjoavat etuja myös vanhojen systeemien uudelleensuunnittelussa. Joskus voidaan nähdä tarve jo olemassa olevien systeemien karakterisointiin ja mallin muodostamiseen [40,77]. Vanhojen systeemien mallien kehittämiseen voidaan käyttää systeemin muuta dokumentointia kuten 3D-malleja, rakennepiirustuksia tai prosessikaavioita. Karakterisoidun systeemin mallin avulla voidaan havainnollistaa systeemin rakennetta, elementtien jäljitettävyyttä sekä rajapintoja alisysteemien välillä. Tämän pohjalta voidaan määrittää systeemin uudelleenkäytettävät osat sekä mahdolliset puutteet jotka vaativat päivitystä.

Lisäksi uusia systeemejä suunniteltaessa, voidaan vanhojen systeemien pohjalta käyttää uudelleen suunnittelukomponentteja sekä vaatimuksia, ja näin vähentää suunnittelukuluja. Eräät mallipohjaisen systeemisuunnittelun menetelmät sisältävät vaiheenaan olemassa olevien järjestelmien karakterisoinnin ja sopivat sellaisenaan erilaisten systeemien uudelleensuunnitteluun. [40]

3.11 Mallipohjaisen systeemisuunnittelun käyttöönotto

Erilaisten mallien ja kaavioiden käyttö on ollut osana myös dokumenttipohjaista systeemisuunnittelua. Osamallit ovat olleet alakeskeisiä tai erikoistuneita tiettyyn systeemin osaan. Mallipohjaisen systeemisuunnittelun tarkoituksena on kuitenkin luoda kokonaisvaltainen ja suunnittelun eri osa-alueita yhdistävä systeemin malli. Kyseisen mallin pohjalta voidaan luoda erilaisia näkökulmia ja dokumentteja systeemin sidosryhmien tarpeisiin. Muutos dokumenttipohjaisesta suunnittelusta mallipohjaiseen tarkoittaa dokumenttien ja analyyttisten mallien hallinnan sijaan systeemin mallin ylläpitoa.

Tässä luvussa tuotujen etujen lisäksi mallipohjainen systeemisuunnittelu vaatii opettelua ja koulutusta, että sen käyttöönotto on mahdollista suunnittelevassa organisaatiossa. Omistautumista se vaatii samalla tavalla kuin minkä tahansa muunkin suunnittelumenetelmän opettelu. Mallipohjaisen systeemisuunnittelun osaajan täytyy hallita MBSE-menetelmä ja jokin mallinnuskieli, jotka mahdollistavat systeemin mallin suunnittelun. Lisäksi täytyy hallita mallinnustyökalun käyttö. Näitä taitoja käytetään hyväksi mallin luomisessa tietylle toimialalle, joten systeemisuunnittelijan täytyy hallita myös perusteet itse toimialasta. Systeemien suunnitteluprojekteihin sisältyy usein muitakin malleja ja näiden integrointi systeemin mallin kanssa on haasteellista. Mallinnus ei ole helppoa, ja hyvästäkin systeemistä voidaan luoda huono malli. Toisaalta myös huonosta systeemistä voi olla erityisen hyvä malli. [40]

Varsinkin suurten systeemien suunnittelussa mukana voi olla erittäin monta mallintajaa. Lisäksi suunnittelijat voivat työskennellä jopa eri puolilta maailmaa. Erityisesti tällöin on tärkeää, että mallinnuksessa on sovittu yhtenäiset säännöt väärinkäsitysten välttämiseksi. Pelkästään mallinnuskielen osaaminen ei riitä. [40]

Organisaation miettiessä mallipohjaisen systeemisuunnittelun käyttöön ottamista täytyy harkita, kumoavatko sen tuomat edut siitä aiheutuvat kustannukset. Harkinnassa pitää ottaa huomioon myös asiakkaat ja muut yhteistyökumppanit, jotka käyttävät suunnittelun pohjalta tehtyjä malleja.

4. AUTOMAATIOJÄRJESTELMÄT

Työssä tehtävässä oppimisympäristön uudistuksessa keskeisessä osassa on tislauskolonniin liittyvä automaatiojärjestelmä sekä prosessin instrumentointi. Tässä kappaleessa on käyty läpi automaation merkitystä sekä yleisimpiä nykyaikaisen teollisuusautomaation suuntauksia, jotka on syytä ottaa huomioon automaation oppimisympäristöä suunniteltaessa

4.1 Automaatio

Automaation merkitys teollisuudessa ja yhteiskunnassa yleensäkin on nykypäivänä suuri. Automaatiolla tavoitellaan esimerkiksi kustannussäästöjä, energiatehokkuutta, parempaa laatua ja toimintavarmuuden parantamista. Erilaisia sovelluskohteita on lukuisia. Automaatiota käytetään esimerkiksi teollisuudessa, taloautomaatiossa, liikenteessä, sulautetuissa järjestelmissä sekä sähköntuotannossa ja jakelussa [52]. Teollisuudessa automaatio jaotellaan yleensä prosessiautomaatioon ja kappaletavara-automaatioon. Kappaletavara-automaatiossa tehdasprosessit ovat usein hyvin nopeita ja automaatiojärjestelmältä sekä siihen liittyvältä verkkoteknologialta vaaditaan suurta nopeutta [53]. Prosessiteollisuudessa automaatioprosessit ovat usein hitaampia, mutta automaatiolta vaaditaan monissa sovelluskohteissa toimintavarmuutta vaarallisissa ja haastavissa ympäristöissä.

4.2 Automaatioverkot

Teolliset automaatioverkot ovat usein huomattavasti useampitasoisia arkkitehtuuriltaan kuin kaupalliset tietoliikenneverkot. Pienetkin automaatio- ja teollisuusverkot ovat usein vähintään kolmi- tai nelitasoisia. [52] Ensimmäinen taso on kenttätaso laitteiden ja ohjausjärjestelmän välillä. Ohjausjärjestelmien ja valvomotason välillä on yksi verkko, joka voi olla vielä yhteydessä toimisto- tai informaatioverkkoon, jonka kautta voi olla yhteys ulkoisiin verkkoihin. Automaatioverkot olivat aikaisemmin usein itsenäisiä verkkokokonaisuuksia, jotka olivat eristettyjä ulkopuoliselta verkkoliikenteeltä. Nykyään automaatiolta vaaditaan yhä useammin mahdollisuus etäkäyttöön esimerkiksi kunnossapitoa ja valvontaa varten. Automaatiojärjestelmien liittäminen ulkoisiin verkkoihin asettaa automaatioverkoille ja automaatiojärjestelmille entistä suurempia tietoturvallisuusvaatimuksia. [52]

Automaatioverkkojen alinta tasoa kutsutaan kenttätasoksi. Tällä tasolla sijaitsevat yleensä erilaiset anturit ja toimilaitteet. Pitkään automaatiojärjestelmissä kenttälaitteet

ovat toimineet 4-20mA standardivirtaviestillä. Jokainen laite erikseen on johdotettu kenttäkaapille, josta kulkee runkokaapeli ristikytkentään. Ristikytkennästä signaalit kulkevat I/O-korteille, joissa analoginen signaali digitalisoidaan ohjausjärjestelmän käyttöön.

Standardivirtaviestin avulla laitteilta pystytään saamaan ainoastaan mittaustietoa tai antamaan laitteille ohjauskäskyjä. Tiedonsiirto ja mikropiirien koon pieneneminen on mahdollistanut älyn kehittämisen itse laitteiden sisään. Älykkäissä kenttälaitteissa on itsessään prosessori ja muistia, jotka mahdollistavat laitteille esimerkiksi diagnostiikkatietojen keräämisen omasta toiminnastaan. Itse tarkoitustaan varten mitattavan suureen lisäksi kenttälaite voi tuottaa myös muita laatusuureita. Älykkäiden kenttälaitteiden sisältämä elektroniikka mahdollistaa myös automaatiojärjestelmän hajauttamisen jopa kenttälaitetasolle asti, koska laitteet voivat itsessään sisältää säätöpiirin. Tämä tuo entistä enemmän joustavuutta automaatiosuunnitteluun ja järjestelmien laajentamiseen [55]. Laitteiden keräämien diagnostiikkatietojen avulla voidaan ennustaa esimerkiksi laitteiden huoltotarpeita ja aikatauluttaa tarvittavia huoltotoimenpiteitä tuotannon kanssa. Näin voidaan välttyä turhilta tuotantokatkoksilta. [56]

Kenttäväylän kehitys alkoi 1980-luvulla. Tarpeena oli kaapeloinnin vähentäminen ja tiedonsiirron mahdollistaminen kenttälaitteiden ja ohjausjärjestelmän välillä. Kenttäväylä tuo monia etuja automaatiojärjestelmien suunnittelussa ja hajauttamisessa. Perinteinen 4-20mA virtaviestilaittein toteutettu instrumentointi vaatii erillisen johdotuksen jokaiselle anturille ja toimilaitteelle ja tarvittavan kaapeloinnin määrä on todella suuri. Kenttäväylän arkkitehtuuri vastaa perinteistä lähiverkkoverkkoarkkitehtuuria, jossa samaan väylään voidaan kytkeä useita eri laitteita. Suunnittelussa ei tarvitse tehdä enää perinteisiä kytkentäluetteloja. Väyläsuunnittelu ei kuitenkaan ole sen helpompaa, ainoastaan suunnittelun luonne muuttuu.

Perinteiset virtaviestilaitteet vaativat liityntäkortin, jossa muodostetaan analogisesta signaalista digitaalinen ja toisinpäin. Isoissa prosesseissa tarvittava liityntäkorttien määrä on suuri. Kenttäväylän tapauksessa kenttälaitteet liitetään samaan kenttäväylään, joka liitetään prosessiaseman väyläkorttiin. Kenttäväylä takaa luotettavan kaksisuuntaisen digitaalisen tiedonsiirron kenttälaitteiden ja ohjausjärjestelmän välillä. Laitteet voivat saada myös tarvittavan käyttöjännitteen väylän kautta.

Etuina kenttäväylän hyväksi ovat alentuneet johdotuskustannukset. Myös I/O-korttien tarve poistuu, koska kenttäväylä liittää kenttälaitteet prosessiasemaan suoraan eikä erillistä I/O:ta ja ristikytkentää tarvita. Myös informaation määrä kentältä lisääntyy ja sen laatu on parempaa. [57]

Kenttäväyläratkaisut tarjoavat kustannusetuja, mutta vanhat virtaviestein toimivat kenttälaitteet eivät suoraan sovellu liitettäväksi kenttäväylään. Laitteet vaativat toimiakseen signaalimuuntimen tai ne joudutaan uusimaan kokonaan [78]. Laitteiden uusiminen tuo

lisäkustannuksia instrumentoinnissa, mutta uudet ja älykkäät laitteet tuovat mukanaan muita etuja, kuten tuotantokatkoksien välttämisen [55,56].

Kenttäväylä mahdollistaa älykkäiden kenttälaitteiden liittämisen, joiden avulla voidaan hankkia prosessitietoa. Lisäksi laitteet voivat tallentaa muita tietoja toiminnastaan. Toimintatietojen avulla laitteet voivat virittää itsensä ja päätellä esimerkiksi vikaantumisesta tai huoltotarpeesta.

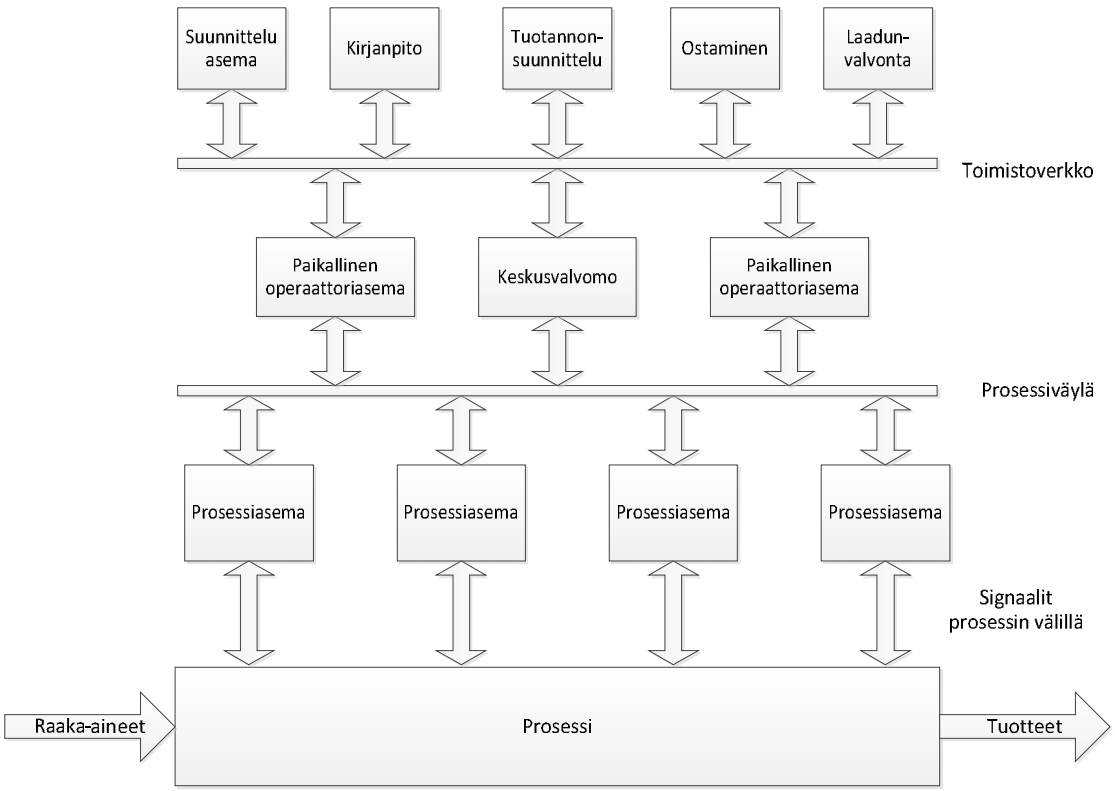
Prosessiautomaation tarpeisiin, joihin liittyy räjähdysvaarallisia tiloja sekä tarve laitteiden virransyöttöön löytyy kaksi kenttäväyläratkaisua Profibus PA ja Foundation Fieldbus H1. Prosessiteollisuuden soveltamiskohteet vaativat kenttäväylältä sopivia käyttöjännitteitä ja standardoituja liittymismenetelmiä kipinöintivaaran poistamiseksi. Tässä diplomityössä on käsitelty ainoastaan kyseisiä Profibus ja Foundation Fieldbus kenttäväyliä, koska ne sisältävät prosessiautomaation käytetyimmät kenttäväylästandardit.

4.3 Ohjausjärjestelmät teollisuudessa

Ohjausjärjestelmät teollisuusautomaatiossa jaotellaan yleensä PLC- tai DCS-pohjaisiksi. PLC (*programmable logic controller*) eli ohjelmoitava logiikka on erillinen tietokone, joka voidaan ohjelmoida hoitamaan loogisia säätöoperaatioita tai sekvenssejä. Alun perin ohjelmoitavat logiikat tulivat kappaletavarateollisuuteen korvaamaan releitä ja ajastimia. Nykyiset ohjelmoitavat logiikat ovat huomattavasti kehittyneempiä ja niillä voidaan suorittaa esimerkiksi säätöpiirejä, aritmeettisia funktioita, matriisilaskuja tai datan prosessointia ja raportointia. [54,58]

Hajautetussa automaatiojärjestelmässä (*distributed control system*, DCS) toiminta on hajautettu pienempiin yksiköihin ja kytkentöihin. Ensimmäiset hajautetut ohjausjärjestelmät ilmestyivät prosessiteollisuuteen 1970-luvulla [58]. Järjestelmä koostuu yleensä useista prosessiasemista, I/O-kokonaisuuksista sekä operaattoriasemista. Kenttälaitteet johdotetaan pienempiin yksiköihin ja säätöpiireihin toiminnallisuuksien mukaan, eikä koko tehtaan toimintaan vaadittavaa kenttäkaapelointia tarvitse vetää yhteen paikkaan keskitetysti. Jokaista yksikköä ohjaa oma prosessiasemansa. Prosessiasemat yhdistetään keskenään prosessiväylillä, jotka ovat nopeaa verkkotekniikkaa. Eri osaprosesseja voidaan valvoa yhdestä tai useammasta valvomosta.

Hajautetuissa automaatiojärjestelmissä on usein panostettu toiminnan redundanttisuuteen, eikä yhden osan vikaantuminen vaaranna koko prosessin toimintaa. Järjestelmissä voidaan kahdentaa esimerkiksi prosessiasemat, I/O-kokonaisuudet sekä tietoliikenneyhteydet. Valvomoita on useita ja prosessia voidaan ohjata esimerkiksi keskusvalvomon vikatilanteessa paikalliselta operaattoriasemalta. Prosessi voidaan liittää myös yrityksen muihin toimintoihin esimerkiksi tuotannonohjausjärjestelmään tai muihin informaatiojärjestelmiin. [58] Yrityksenlaajuisesta hajautetusta ohjausjärjestelmästä on esimerkki kuvassa 31.



Kuva 31. *Yrityksenlaajuinen DCS, mukaillen [54].*

Prosessiasemat olivat pitkään valmistajien omaa laitteistoa. PC tietokoneiden teho on kehittynyt ja käyttöjärjestelmät mahdollistavat reaaliaikatoiminnot. Tämän myötä ovat nykyään usein räätälöityjä pc-tietokoneita teollisuuden vaativiin kohteisiin tarkoitetussa koteloinnissa. Yksi prosessiasema pystyy usein huolehtimaan suuresta kenttälaitteiden määrästä. [54]

Eräiden kenttäväyläratkaisujen avulla prosessin ohjaus voidaan hajauttaa aina kenttälaitteiden hoidettavaksi asti. [58]

Hajautettujen järjestelmien ja PLC:n käyttötarkoitus ei jakaudu enää niin selvästi, kuin niiden alkuperäiset käyttökohteet olivat ajateltu. PLC-ratkaisuja voidaan käyttää prosessien ohjauksessa ja hajautettuja ratkaisuja kappaletavara-automaation sovelluksissa.

4.4 PROFIBUS

Euroopassa käytetyimmäksi kenttäväyläksi on noussut PROFIBUS (Process Field Bus). Kyseisestä väylästandardista on nykyään käytössä kaksi versiota Profibus DP (*decentralized peripherals*) ja Profibus PA (*process automation*). Profibus DP on tarkoitettu tehdasautomaation rajamittauksiin ja toimilaiteohjauksiin. Profibus PA on puolestaan eri-

tyisesti prosessiautomaation tarpeisiin tarkoitettu kenttäväyläratkaisu. Molemmat Profibus väylätyypeistä käyttävät samaa tietoliikenneprotokollaa.

Liikennöinti Profibus-väylässä tapahtuu isäntä-orja periaatteella. Väylän isäntälaitteena voi toimia esimerkiksi prosessiasema. Isäntälaite kysyy tietoja orjalaitteilta. Orjalaite ei voi lähettää tietoa väylään ennen kuin isäntälaite antaa luvan. Profibus väylässä voi olla kahden tyyppisiä isäntälaitteita. Luokan yksi isäntä on normaalisti säädin, kuten PLC tai prosessiasema. Luokan kaksi isäntälaite on työasema, jolla voidaan tarkkailla verkon tilaa ja suorittaa vianmääritystä. Väylässä voi olla yksi tai useampia isäntälaitteita. Isäntälaitteet saavat vuorollaan kysyä tietoja orjalaitteilta. Väylä on valtuudenvälitysverkko, jossa isäntälaitteet saavat vuorottain kysyä ja lähettää tietoa orjalaitteille vuorotellen. Kaikkien laitteiden käytettyä vuoronsa alkaa kierros uudelleen. [53] Koska kenttälaitteet tarvitsevat luvan isäntälaitteelta tiedon lähettämiseen, eivät kenttälaitteet voi toimia itsenäisesti Profibus-väylässä.

Profibus väylässä voi kulkea kahta erilaista viestityyppiä: aikakriittisiä ja ei-aikakriittisiä viestejä. Viestit eroavat toisistaan sen ajan suhteen miten ne välittyvät väylässä. Aikakriittiset viestit kuuluvat syklisiin palveluihin. Ne suoritetaan väylän jokaisella kierroksella. Normaalisti aikakriittiset viestit ovat säätöpiirien tarvitsemaa input- ja outputtietoa. Ei-aikakriittiset viestit ovat asyklisiä palveluja ja niiden suoritus voi kestää monta kierrosta. Normaalisti tällä tavalla siirretty data on tarkoitettu diagnostiikkatietojen ja parametrien välitykseen, joita automaatiojärjestelmää valvova käyttäjä tarvitsee. [53]

Kappaletavara-automaatiossa eri automaatiolaitteiden välinen tiedonsiirto vaatii suurta nopeutta. Profibus DP muodostaa koko Profibus-standardin perustan. Se toimii nopeiden laitteiden kuten rajakytkimien ja taajuusmuuttajien kenttäväylänä. Toinen käyttökohde sille on yhdistää Profibus PA-segmentit automaatiojärjestelmään tai toimia erilaisten I/O-kokonaisuuksien liittämiseen prosessissa. Profibus DP:n fyysisenä kerroksena voi toimia RS-485, valokuitu tai langaton tiedonsiirtotapa. Näistä RS-485 on useimmiten käytetty hyvän tiedonsiirtokykynsä takia. Lisäksi se sietää hyvin häiriöitä teollisessa ympäristössä. [53]

Prosessiautomaatiossa kenttäväylään liitettävät laitteet ovat yleensä mittauslähettimiä ja toimilaitteita. Usein laitteet tarvitsevat erillisen tehonsyötön. Prosessiteollisuudessa on sovelluksia, joissa kipinöintiä ei voida sallia laitetilassa räjähdysvaaran vuoksi. Profibus PA kenttäväylä pystyy tehonsyöttöön laitteille. Tehonsyöttö on rajoitettu kuitenkin niin, ettei kipinöintiä tai laitteiden ja johtimien liiallista lämpiämistä pääse tapahtumaan [53].

Profibus PA fyysisessä kerroksessa käytetään IEC 61185-2 standardin mukaista MBP-menetelmää (*Manchester Encoded Bus Powered*). Väylän jännite on vähintään 9 V ja enintään 32 V. Väylän tietoliikenteen siirtonopeus on kiinteä 31.25 kbit/s. Profibus PA-

väylä liitetään ohjausjärjestelmään Profibus DP-väylän kautta, johon se liittyy DP/PA-kytkimellä.

4.5 Foundation Fieldbus

Foundation Fieldbus on täysin digitaalinen, kahdensuuntaiseen liikenteeseen tarkoitettu kenttäväyläratkaisu. Se pitää sisällään H1 ja HSE konfiguraatiot. H1 on matalan tiedonsiirtonopeuden väylä, joka mahdollistaa tehonsyötön kenttälaitteille. Se on tarkoitettu lähinnä jatkuvan prosessinohjauksen sovelluskohteisiin. H1-väylän korkein tiedonsiirtonopeus on Profibus PA-väylän tavoin 31,25 kb/s ja se noudattaa samaa MBP protokollaa fyysisellä kerroksella.

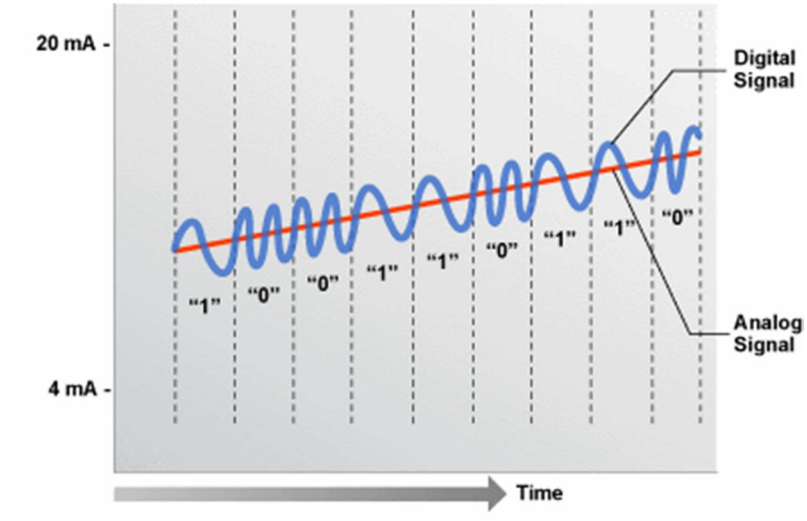
Foundation Fieldbus HSE perustuu Ethernet-tekniikkaan ja mahdollistaa 100Mbit/s tiedonsiirtonopeuden. HSE on tarkoitettu yhdistämään ohjausjärjestelmiä ja H1-segmenttejä. [58] Foundation Fieldbus muistuttaa ominaisuuksiltaan paljon Profibus teknologiaa. Foundation Fieldbus H1 kenttäväylän erikoisuutena on, että sen avulla mahdollistuu myös säätöpiirien jakaminen kenttälaitteiden suorittamiksi.

4.6 HART

HART protokolla kehitettiin tarpeeseen saada ja välittää informaatioita kenttälaitteiden ja ohjausjärjestelmän välillä muuttamatta perinteisiä milliampeeriviesteihin perustuvia johdotuksia. Protokollan kehitti Rosemount 1980-luvun puolivälissä ja se tuli avoimeksi protokollaksi vuonna 1993 [53]. HART ei ole varsinainen kenttäväyläprotokolla, mutta se mahdollistaa digitaalisen tiedonsiirron kenttälaitteilta sekä tietyssä tapauksessa useamman kenttälaitteen kytkemisen samaan johtimeen.

HART noudattaa tietoliikenteen 7 kerroksista OSI-mallia, joista se määritellään kerroksille 1,2,3,4 ja 7. [58]

Ensimmäinen OSI-mallin kerros eli fyysinen kerros perustuu HART-protokollassa Bell 202 kommunikaatiostandardin taajuusmodulointimenetelmään. Kyseisessä menetelmässä perinteisen 4-20mA standardivirtaviestin päälle moduloidaan digitaalinen signaali, joka on esitetty kuvassa 32. HART-signaali siirretään tavallisen analogisen virtasignaalin päällä niin, ettei analoginen informaatio häiriinny. HART-signaalissa 1200 Hz taajuus vastaa digitaalista numeroa yksi ja 2200 Hz taajuus digitaalista nollaa. HART-protokollan tiedonsiirtonopeus on 1200 bittiä sekunnissa.



Kuva 32. *HART-signaali [59].*

OSI-mallin toisella eli datayhteyskerroksella HART-protokollassa käytetään isäntä/orja-protokollaa. Tällöin orjalaitteet vastaavat ainoastaan isäntälaitteen kysyessä. HART-protokolla mahdollistaa kahden isäntälaitteen liittämisen samaan johtimeen. Ensimmäinen isäntälaite voi olla esimerkiksi ohjausjärjestelmä, toinen HART-konfigurointilaite [60].

Kolmas kerros eli verkkokerros selittää reitittämisen ja kommunikaation. Johtimissa käytettynä HART-protokollalla ei ole varsinaista verkkokerrosta. Kuljetuskerros eli OSI-mallin neljäs kerros pitää huolen, että HART-viesti kulkeutuu laitteiden välillä [58]. OSI-mallin viimeisessä seitsemännessä kerroksessa eli sovelluskerroksessa HART-protokollassa määritetään erilaiset HART-komennot.

HART-protokolla mahdollistaa kenttälaitteiden konfiguroimisen erillisten kannettavien laitteiden avulla. Lisäksi on mahdollista lukea laitteiden diagnostiikkatietoja, ylimääräisiä mittauksia, laitteen tilaa tai suorittaa vianetsintää [59]. Usein HART-laitteet ovat kiinni järjestelmässä 4-20mA I/O-kortin kautta ja digitaalista tiedonsiirtoa käytetään ainoastaan laitteita konfiguroitaessa [53].

Analogista signaalia käytettäessä ainoastaan yksi laite voi kommunikoida. Kahden laitteen syöttäessä virtaa linjalle samaan aikaan lopullinen virtasignaali olisi virheellinen molemmille laitteille [61]. HART-protokolla mahdollistaa myös useamman laitteen yhtäaikaiseen kommunikointiin. Tätä kutsutaan multidrop-kommunikoinniksi. Kenttälaitteiden tarvitsema virta voidaan tuoda johtimen avulla tai laitteet voidaan yksittäin varustaa tehonsyötöllä. Multidrop-kommunikoinnissa laitteet voivat viestiä ainoastaan digitaalisella HART-signaalilla [60].

4.6.1 WirelessHART

Langatonta laitteistoa erilaisissa prosesseissa on usein käytetty ainoastaan järjestelmän tarkkailuun. Itse prosessin säätäminen on toteutettu perinteisesti johtimia pitkin. Langatonta tekniikkaa on käytetty ylimääräisen datan keräämiseen.

Käytettäessä langatonta tekniikkaa itse prosessin säätämiseen kohdataan erilaisia haasteita. Langattomien verkkojen tapauksessa tieto kulkee ilmassa radioaaltoina. Tiedonsiirtoon ilmassa voi väliaikaisesti vaikuttaa esimerkiksi sää, liikkuvat ihmiset ja esineet sekä muut langattomat signaalit tai sähkömagneettiset häiriöt. Nämä kaikki seikat voivat vaikuttaa tiedonsiirron oikea-aikaisuuteen ja sitä kautta prosessin säätöön. [62]

Langattoman tiedonsiirron tarpeisiin prosesseissa on kehitetty HART-protokollaa käyttävä langaton siirtotapa. WirelessHART toimii IEEE 802.15.4-2006 standardin mukaan ja toimii yleisellä 2,4 GHz taajuudella.

WirelessHART-verkon topologia on niin sanottu mesh-verkko, jossa jokainen laite voi toimia reitittimenä. Koska HART-viesti voidaan lähettää WirelessHART-verkossa mitä tahansa reittiä pitkin, mesh-verkon topologia auttaa pitämään yllä verkon redundanttisuutta. WirelessHART verkko yhdistetään prosessinohjaukseen WirelessHART-gatewayn välityksellä. Gatewayn ja prosessinohjauksen välinen yhteys tapahtuu esimerkiksi Modbus tai HART-IP protokollien avulla.

Kenttälaitteet voivat olla valmiiksi sisäänrakennettuja WirelessHART yhteensopiviksi. Myös tavalliset HART-protokollaa käyttävät kenttälaitteet saadaan liitettyä WirelessHART-verkkoon erillisen adapterin avulla. Useat adapterit toimivat paristolla, joka voi kestää yleensä useita vuosia. Kenttälaitteet voivat saada tarvittaessa myös tehonsyöttönsä adapterilta, joten erillistä virtajohdotustakaan ei näissä tapauksissa tarvita. Käytettäessä pelkästään langattomia antureita vältytään myös kenttäkoteloiden sekä I/O-yksiköiden hankkimiskustannuksilta.

Langatonta tiedonsiirtoa automaatioratkaisuissa voidaan tarvita varsinkin, jos kenttälaitteet sijaitsevat hankalassa paikassa tai välimatka laitteille on niin pitkä, ettei ole järkevää tehdä perinteistä kaapelointia. Muita mahdollisia tilanteita, joissa WirelessHART-teknologiaa voi käyttää hyväkseen, ovat esimerkiksi uudet mittauspisteet prosessissa, joiden kaapelointi olisi hankalaa sijainnin tai tilanpuutteen vuoksi. Teknologiaa voidaan käyttää myös väliaikaisissa prosessikokeiluissa tai pilottihankkeissa ilman fyysisen kaapeloinnin tarvetta. Nykypäivänä monet HART-laitteet ovat 4-20mA I/O-kortteihin kytkettynä ilman mahdollisuutta diagnostiikkaan. WirelessHART-teknologian integrointi mahdollistaa näissä tapauksissa HART-laitteiden diagnostiikan lukemisen ja laitteiden konfiguroinnin. [59]

Langattomaan tiedonsiirtoon sopivat kenttälaitteet ovat huomattavasti kalliimpia perinteisiin verrattuna. Myös laitteisiin liitettävä WirelessHART-adapteri tulee usein normaa-

lia johdotusta kalliimmaksi. Langaton vaihtoehto onkin järkevää toteuttaa silloin, kun johdotusmatkat ovat pitkiä, ne tulisivat kalliiksi tai johdotus olisi muuten vaikeaa toteuttaa esimerkiksi järjestelmän laajennuksissa tilanpuutteen vuoksi.

4.7 OPC

Automaatiojärjestelmät joudutaan usein kokoamaan eri laitetoimittajien kesken. Automaatiolaitteiden kesken tapahtuva kommunikaatio vaatii yleensä ainutkertaisen määrittämisen eri valmistajien laitteiden välille. OPC teknologia on kehitetty yhdistäväksi standardiksi tiedonsiirtoon eri valmistajien laitteiden välillä. Teknologia mahdollistaa myös erilaisten informaatiojärjestelmien integroimisen automaatiojärjestelmiin, joka on nykyään ehdoton vaatimus suurissa prosesseissa.

OPC lyhenne tulee alun perin sanoista OLE for Process Control. OLE (*object linking and embedding*) on Microsoftin lanseeraama tiedonsiirtoteknologia. Nykyisin lyhenteellä tarkoitetaan sanoja Openness, Productivity and Collaboration.

Ensimmäinen OPC spesifikaatio oli OPC Data Access, joka käsittelee reaaliaikaisen datan siirtämistä. Myöhemmin OPC laajeni käsittelemään hälytysdataa (*Alarms&Events*, AE) ja historiadataa (*Historical Data Access*, HDA) koskevia spesifikaatioita. [55] Nykyään kyseiset spesifikaatiot kuuluvat OPC Classic määrityksen alle.

Ensimmäisissä OPC-spesifikaatioissa oli ongelmana, että ne olivat Microsoftin DCOM-teknologiaan pohjautuvia. Rajoituksena oli muuan muassa se, että asiakaan ja palvelimen täytyi olla samassa intranetissä. Lisäksi toiminta oli mahdollista ainoastaan Windows-ympäristössä. [55] Tietoturvallisuus ei ollut myöskään nykyisten standardien tasolla.

Vuonna 2008 OPC julkaisi OPC UA (*Unfied Architecture*) alustariippumattoman arkkitehtuurin, joka yhdisti aiemmat spesifikaatiot. Teknologia toimii erilaisilla laitteistoalustoilla kuten perinteisellä PC-laitteistolla, pilvipalveluissa, ohjelmoitavissa logiikoissa sekä mikrokontrollereissa. Samalla uusi teknologia teki automaatiojärjestelmät riippumattomaksi Windows-alustasta, koska arkkitehtuurissa liikennöinti ei enää perustu DCOM-teknologiaan. OPC UA mahdollistaa toiminnan myös Linux, Apple OSX ja Android käyttöjärjestelmissä perinteisen Microsoft Windowsin lisäksi. OPC-teknologian vahvuutena on nykystandardin myötä hyvä skaalautuvuus, laajennettavuus ja parantunut tietoturva. [66]

5. TUTKIMUKSEN LÄHTÖTILANNE

Uudistettava oppimisympäristö rakentuu Systeemitekniikan laitoksen binääritislauskolonnin sekä siihen liittyvän prosessi- ja automaatiolaitteiston ympärille. Tislauskolonni on valmistunut vuonna 1983 pilottiprojektina diplomityön myötä [4]. Kolonni hankittiin hyödynnettäväksi opetus- ja tutkimuskäytössä. Tislausprosessi ja siihen liittyvä automaatiojärjestelmä on ollut hyvä ympäristö tutustuttaa opiskelijat teollisten prosessien hallintaan sekä niihin liittyvään automaatioon ja säätötekniikkaan.

5.1 Tislauksesta yleisesti

Tislaus on yksi prosessiteollisuuden tärkeimmistä operaatioista. Tislaus perustuu tislattavien aineiden eri haihtuvuuksiin eli höyrynpaineisiin tietyssä lämpötilassa. Kuumennettaessa aineseosta muodostuva höyry sisältää suuremman pitoisuuden ainetta, jolla on suurempi haihtuvuus eli sen höyrynpaine on suurempi. [63] Tislausprosessi on niin kutsuttu yksikköprosessi eikä siinä tapahdu kemiallista reaktiota. Prosessissa tapahtuva fyysinen reaktio on faasimuutos. Tasapainotilanteessa aineiden pitoisuudet kaasufaasissa ovat erilaiset kuin nestefaasissa [64].

Kaupallisesti tislauksella on useita tärkeitä sovelluskohteita. Eräitä tärkeimpiä ovat: raaka-öljyn erottaminen eri öljyjalosteiksi, veden puhdistaminen ja suolan poistaminen, ilman eri komponenttien erottaminen sekä käymisen tuotteena syntyneiden alkoholipitoisten nesteiden tislaus [65]. Monimuotoisen soveltamisen takia tislausprosessiin liittyvän teorian ja teknologian opettaminen prosessien hallintaan liittyen on tärkeätä. Binääritislausprosessissa eroteltavassa aineliuoksessa on kahta osa-ainetta. Teollisuudessa tislausprosessit ovat usein monikomponenttisia, eli samassa prosessissa lähtöaineesta erotellaan useampia ainesosia. Binääriprosessi on kuitenkin yksinkertainen tapa tutkia ja opiskella tislausprosesseihin liittyvää teoriaa ja tekniikkaa.

Tislausprosessissa tärkein laadun tavoite on tuottaa hyvälaatuista lopputuotetta energiatehokkaasti. Systeemitekniikan laitoksen kolonnissa tislataan etanoli-vesi-liuosta. Prosessissa tavoitteena saavuttaa mahdollisimman tasalaatuinen korkeaetanolipitoinen tisle lopputuotteeksi. Laitoksen prosessissa ei ole pitoisuuden mittausta, joten tuotteen pitoisuutta voidaan säätää esimerkiksi huipun lämpötilaa ja painetta säätämällä, käyttäen hyväksi höyry-neste käyriä. Tislausprosessin onnistumista voidaan tarkastella mittaamalla lopputuotteen ominaispainoa ja vertailla vastaako se prosessin mittausten ja höyry-neste käyrien osoittamaan lopputulokseen.

5.2 Systeemitekniikan laitoksen tislauskolonnin prosessilaitteet

Tislauslaitteistoon kuuluu tislauskolonni, kaksi 1m^3 säiliötä syötteelle, yksi 0,5m^3 säiliö tisleelle sekä yksi 0,5 m^3 säiliö pohjatuotteelle. Tislauskolonnin korkeus on 5,7 m ja sen halkaisija 0,3 m. Kolonnissa on 12 välipohjaa, jotka tehostavat tislausprosessia. Välipohjat ovat systeemitekniikan laitoksen tislauskolonnissa kellotyyppisiä.

Prosessin putkistot jakautuvat neljään osaan: prosessineste-, höyry-, kylmävesi- ja huohotusputkiin. Prosessinesteputkistoon kuuluvat syöttö-, alitevirtaus- , tisle- ja palauteputket. Höyryputkiston tehtävänä on johtaa höyrykattilan tuottama höyry pohjakiehuttimelle sekä syötevirtauksen lämmönsiirtimille. Näiden jälkeen lauhtunut höyry kulkeutuu lauhdesäiliöön. Kylmävesiputkiston kautta johdetaan vesijohtoverkon kylmä vesi lauhduttimelle, apulauhduttimelle sekä tisleen jäähdyttimelle, joiden kautta vesi virtaa lopuksi viemäriin. Huohotusputkiston tehtävänä on tasata prosessin säiliöiden välistä painetta. Se yhdistää säiliöiden ja apulauhduttimen ilmatilat ulkoilmaan. Huohotusputkistolla on oma jäähdyttimensä, jonka avulla estetään itse tisleen haihtuminen ulkoilmaan.

Prosessinesteen siirtämisestä huolehtii kolme pumppua: yksi syöttövirtausta varten, yksi tislevirtausta varten ja yksi alitevirtausta varten. Pumput ovat vakiokierrospumppuja ja virtausten säätö toteutetaan säätöventtiileillä.

Tislaukseen tarvittava lämpö tuotetaan Zeta Z-100 MS-höyrynkehittimellä. Se pystyy tuottamaan höyryä 65 kW maksimiteholla. Kyseisessä höyrynkehittimessä vesi kiehutetaan elektrodeilla, jotka johtavat sähkövirran veden läpi. Kattilassa täytyy olla riittävästi vettä, että virranjohtavuus säilyy. Höyrynkehittimen automatiikka pitää huolta, että höyrykattilan vedenpinta pysyy tarpeeksi korkealla. Kiehutusveden virranjohtavuutta voidaan kasvattaa lisäämällä siihen trinatriumfosfaattia. Sen annostelussa täytyy kuitenkin olla tarkkana, ettei kattilassa pääse tapahtumaan oikosulkua. Höyrystä saatava lämpö siirretään tislausprosessiin pohjakiehuttimena toimivan pystyputkilämmönsiirtimen avulla. Syöttönesteen lämmitykseen, prosessihöyryn lauhdutukseen ja tisleen jäähdytykseen käytetään viittä putkilämmönsiirrintä.

Systeemitekniikan laitoksen tislauskolonni sijaitsee Tampereen teknillisen yliopiston sähkötalon laboratoriohallissa. Laboratoriohallin lämpötila riippuu vahvasti ulkoilman lämpötilasta. Kolonni on eristämätön ja osaksi siksi prosessin ylösajo on hidasta. Laboratoriotilan ulkoiset lämpötilanvaihtelut vaikuttavat lisäksi merkittävästi prosessin käyttäytymiseen. [5]

5.3 Laitoksen tislausprosessin instrumentointi

Prosessin instrumentointi on pääasiassa alkuperäistä. Prosessissa on 25 Pt100 lämpötila-anturia. Pohjakiehuttimelle menevän höyryn lämpötilan mittaukseen käytettävän anturin aikavakio on kolme sekuntia. Muiden prosessissa käytettävien lämpötila-antureiden aikavakio on n.15 sekuntia. Lämpötilamittausten lähettimet sijaitsevat laitekaapissa. Lähes kaikki lämpötila-anturit ja lähettimet ovat alkuperäisiä.

Prosessinesteen virtausta mitataan neljällä Litre-Meterin valmistamalla virtausmittarilla. Kyseisissä virtausmittareissa virtauskammiossa pyörivän roottorin ferriittisauvat indusoivat anturille pulssijonon, joka muutetaan lähettimessä standardiviestiksi. Syöttö- ja alitevirtauksen virtausmittareiden anturit ovat tyyppiä LM45. Niiden mittausalue on 0,04..6,4 l/min. Tisle- ja palautevirtauksen tyypin LM24 antureiden mittausalue on 0,03..4,3 l/min.

Akun ja kolonnin pohjan pinnankorkeutta on mitattu Valmetin Diff-El DP paine-erolähettimellä. Pohjan pinnankorkeuden mittaus on viritetty 10 % etanoli-vesi-liuoksella 0..0,70m korkeusalueelle. Akun pinnankorkeuden mittaus on viritetty 0..0,30m alueelle 80 % etanoli-vesi-liuokselle. Kolonnin huipun painetta on mitattu Valmetin Press-el AP painelähettimellä. Kummankin tyypin lähettimillä on sama toimintaperiaate. Paineen mittaus tapahtuu differentiaalikondensaattorin kapasitanssin muutosta mittaamalla. Myös kolonnin alimmalla pohjalla on Press-el AP painelähetin. Pohjan paineen havaittiin kuitenkin olevan lähes identtinen huipun paineen kanssa, joten pohjan paineen mittausta ei ole käytetty prosessin säädössä.

Paine-ero lähettimien mittauksen on havaittu ryömivän, eli ne ovat vaatineet usein manuaalista virittämistä oikean arvon saamiseksi. Kaikki painetta mittaavat lähettimet ovat lisäksi noin kolmenkymmenen vuoden takaa. Mittaukset ovat toimivia, mutta teknologia vanhentunutta.

Prosessia säädetään seitsemällä säätöventtiilillä, joita kutakin ohjataan pneumaattisella toimilaitteella. Näistä toimilaitteista kuudessa on sähköpneumaattinen asennoitin, jotka ovat pääosin alkuperäisiä. Syöttövirtauksen säätöön tarkoitetussa venttiilissä on digitaalinen Metso Neles ND9000 asennoitin. Säätöventtiilien lisäksi prosessissa on useita käsiventtiileitä.

Eräät alkuperäisen järjestelmän säätöventtiilit olivat ylimitoitettuja, jonka takia prosessin hallinta oli vaikeata. Keskeisessä asemassa oleva lämmityshöyryn virtausta säätävä venttiili vaihdettiin uuteen vuonna 1995 diplomityöprojektin yhteydessä. Tämän vaihdoksen myötä koko prosessin hallittavuus kasvoi selvästi. [7] Syöttönesteen esilämmitykseen käytettävää höyrynsäätöventtiiliä ei ole pystytty käyttämään ollenkaan. Sen väärästä mitoituksesta johtuen prosessin hallitseminen on ollut mahdotonta venttiiliä käytettäessä.

Alitevirtauksen säätöön käytetty venttiili on ollut erään harjoituksen demokäytössä ja sitä on viritetty asennoittimessa sijaitsevalla viritysruuvilla. Asennoittin on sen verran kulunut, että koko venttiilin säädettävyys on huono.

5.4 Tislausprosessin automaatiojärjestelmä

Ensimmäisenä automaatiojärjestelmänä vuonna 1983 valmistuneessa tislausprosessissa toimi Valmetin Damatic 60, joka korvattiin samana vuonna Damatic 64 pienvalvomolla [4]. Seuraavaksi automaatiojärjestelmäksi hankittiin Damatic Classic automaatiojärjestelmä, jota seurasi vielä Damatic XD automaatiojärjestelmä. Vuonna 2000 prosessinohjauksessa otettiin käyttöön Metso DNA automaatiojärjestelmä, joka on päivitetty viimeksi vuonna 2007.

Ennen tässä diplomityössä tehtävää oppimisympäristön uudistusta tislausprosessin automaatio-ohjelmiston lisäksi itse prosessiasema on toiminut Windows-pohjaisella PC-tietokoneella. Tämä ei ole yleinen toteutustapa vaativissa teollisuusprosesseissa. Kyseinen toteutus voi olla epävarma, koska tietokoneella pyörii muitakin ohjelmistoja kuin pelkkä prosessinohjaus.

Automaatiojärjestelmään liittyvä laitekaapelointi ja sen ristikytkentä on toteutettu epästandardilla tavalla. Kaikki kenttälaitteet on johdotettu suoraan laitekaapin ristikytkentään. Ristikytkennästä signaalit ovat aiemmin johdotettu lattakaapeleilla I/O-korteille. Viimeisimmän laitteistopäivityksen myötä lattakaapelit ovat purettu osikseen ja johdotettu I/O-korteille epäselvällä tavalla. Nykyinen automaatiojärjestelmän kaapelointitoteutus ei ole opetuksellisesti järkevä.

Suurin osa tislausprosessin tämänhetkisestä automaatiolaitteistosta ja instrumentaatiosta on toimivaa, mutta kuitenkin osaltaan vanhentunutta ja esimerkiksi varaosasaatavuus laitteisiin on heikko. Teollisuuden toimitusprojekteissa joudutaan usein integroimaan uutta ja vanhaa automaatiolaitteistoa ja teknologiaa. Siksi on opetuksellisesti järkevää, että prosessin oppimisympäristö sisältää tulvaisuudessakin myös vanhempaa laitteistoa, vaikka instrumentointia päivitettäisiinkin. Hyvästä toimintakyvystä täytyy kuitenkin varmistua.

5.5 Tutkimus laitoksen tislausprosessiin liittyen

Tislauskolonnia on käytetty tutkimuskäytössä pääasiassa opinnäytetöiden yhteydessä. Kolonni valmistui pilottiprojektina 1983 diplomityön myötä [4]. Vuonna 1989 Pullinen kehitti diplomityössään tislauskolonnin prosessiin erillisellä tietokoneella ajettavan mittaus- ja säätöohjelmiston helpottamaan prosessin hallintaa ja säätöä [5]. Myöhemmin vuonna 1993 Pullinen tutki lisensiaatintyössään erilaisia monimuuttujasäätömenetelmiä kolonnin tislausprosessin hallinnassa [6].

Prosessin säätöä sumealla logiikalla tutkittiin Pitkäniemen diplomityössä, joka valmistui vuonna 1995. Kyseisessä diplomityössä kehitettiin silloisen Valmet Damatic XD perusautomaatiojärjestelmän rinnalle sumea säätäjä. Sumea säädin toteutettiin Omronin ohjelmoitavalla logiikalla ja siihen liitettävällä sumean säädön yksiköllä. Sumealla säädöllä saatiin jonkin verran parantuneita tuloksia tislausprosessin ohjauksessa. [7]

Tämän diplomityön kanssa samaan aikaan tehtiin TTY:n arkkitehtuurin laitoksella Suomen Yliopistokiinteistöt Oy:n teettämää tutkimusta yhteiskäyttöisten laboratoriotilojen käytöstä. Systeemitekniikan laitoksen tislauskolonnin laboratorio oli yhtenä tutkimuksen kohteena. Tämän tutkimuksen myötä laboratoriota käyttäneille opiskelijoille järjestettiin kysely, jonka tulokset toimivat tässä diplomityössä yhtenä lähteenä vaatimuksille oppimisympäristön päivityksessä.

Nykyisen oppimisympäristön uudistuksen jälkeen kolonnilaboratoriosta tulee osa TUT-CyberLabs tietoturvallisuuden tutkimus- ja opetusympäristöä.

5.6 Opetus tislausprosessilaboratoriossa

Tislauskolonniin liittyen on järjestetty vuosien myötä useiden kurssien laboratorioharjoituksia. Teknologian vaihtuessa ja kurssisisältöjen muuttuessa myös oppimisympäristössä suoritettavat harjoitukset ovat muuttuneet. Tässä diplomityössä tislauslaboratoriossa suoritettavien nykyisten laboratorioharjoitusten perustalta muodostetut käyttötapaukset kuvaavat toiminnallisia vaatimuksia, joiden perusteella oppimisympäristön tarvittavat uudistukset voidaan tehdä. Oppimisympäristössä suoritettavat laboratorioharjoitukset on jaettu kolmeen harjoitukseen eri kurssien yhteydessä.

Ensimmäisen kerran opiskelijat tutustuvat tislausprosessiin kurssilla ASE-1130 Automaatio. Tässä harjoituksessa opiskelijat ovat suorittaneet 45 minuuttia kestävän demoharjoituksen noin kymmenen henkilön ryhmissä. Demoharjoituksessa on tutustuttu tislauksen periaatteeseen ja säätöön. Lisäksi on perehdytty pintapuolisesti tislauskolonnilaitteistoon, instrumentointiin sekä kolonnin Metso DNA-automaatiojärjestelmään opettajan avustuksella. Tämän harjoituksen suorittajat ovat olleet yleensä ensimmäisen tai toisen vuoden opiskelijoita. Harjoitusta on edeltänyt noin 45 minuutin luento, jossa harjoituksen ja tislausprosessin perusperiaatteita on selvitetty etukäteen. Sama demoharjoitus on järjestetty myös kurssin ASE-1251 Järjestelmien ohjaus yhteydessä. [67]

Seuraava harjoitus tislauskolonnin oppimisympäristössä kurssin on ASE-2170 Automaatiojärjestelmät ja –suunnittelu. Harjoituksen aikana opiskelijat ovat lisäksi konfiguroineet automaatiosuunnitteluohjelmalla automaatiomoduulin ja valvomonäytön, jonka avulla prosessin syöttövirtauksen ohjausta on demonstroitu. Aiemmin harjoitukseen on liittynyt instrumentointiin tarkemmin tutustuminen ACI-31040 Automaatiolaitteet ja verkot kurssin yhteydessä. Kolonnin instrumentointiin liittyvä harjoitusosio on poistu-

nut nykyisestä toteutuksesta. Tämän harjoituksen suorittaneet ovat olleet yleensä toisen vuosikurssin opiskelijoita. [68,69]

Itse oikean tislausprosessin laboratorioharjoituksen opiskelijat ovat suorittaneet kurssilla ASE-4030 Prosessien hallinnan sovellukset. Laboratorioharjoitus on ollut noin kahden tunnin mittainen. Harjoituksen alussa on käyty läpi laboratorioharjoitukseen vaadittava esiselostus. Itse prosessin tislausharjoituksessa opiskelijat ovat suorittaneet prosessissa askelvastekokeen. Tislausprosessissa tasapainotilan löytämisessä menee aikaa, joten opiskelijoilla on mahdollisuus tutustua harjoituksen aikana kolonnin laitteistoon, instrumentointiin sekä automaatiojärjestelmään. Laboratorioharjoituksessa on kerätty lisäksi dataa, jonka avulla prosessia on myöhemmin mallinnettu simuloinnin avulla. Tämän harjoituksen suorittajat ovat yleensä 3-5 vuosikurssien opiskelijoita.

Lisäksi kurssin ASE-7610 Automaation turvallisuus yhteydessä on arvioitu kolonnin ja sen ympäristöön liittyviä turvallisuusasioita. Uudistuksen myötä oppimisympäristössä on tarkoitus lisätä automaation turvallisuuteen liittyviä harjoituksia. Oppimisympäristössä järjestettävät nykyiset kurssiharjoitukset on kerätty taulukkoon 3.

Taulukko 3. *Oppimisympäristön nykyiset kurssiharjoitukset.*

Kurssin tunnus	Kurssin nimi	Harjoituksen sisältö
ASE-1130	Automaatio	Demoharjoitus
ASE-1251	Järjestelmien ohjaus	Demoharjoitus
ASE-2170	Automaatiojärjestelmät- ja suunnittelu	Automaatiosuunnitteluharjoitus
ASE-4030	Prosessien hallinnan sovellukset	Tislausharjoitus
ASE-7610	Automaation turvallisuus	Automaation turvallisuusasiat

5.7 Oppimisympäristön fyysiset tilat ja työskentely-ympäristö

Tislauskolonnilaitteisto sijaitsee Tampereen teknillisen yliopiston sähkötalon laboratoriohallissa. Kolonnilaboratorio on eristetty muusta hallista sermeillä. Tislauskolonnin prosessilaitteisto on sijoiteltu kahteen kerrokseen. Laboratoriohallissa sijaitsee yliopiston muidenkin laitosten laboratoriotiloja ja hallissa voi olla välillä meluisaa. Lisäksi itse tislausprosessi aiheuttaa melua lähinnä höyrynkehittimen, pumppujen ja paineilmalaitteiden takia.

Tislauskolonniympäristön laitteisto ja tilat ovat päässeet ajan myötä osittain huonoon kuntoon. Ympäristö on muodostunut ajan myötä sekavaksi ja siellä on paljon käyttämätöntä tavaraa. Vuosien saatossa laboratoriossa järjestettyjen eri kurssien harjoitusten materiaalit ovat varastoitu laboratoriotilaan edelleen, vaikkei niillä ole enää varsinaista käyttöä. Järjestelmän suunnitteluaseman ja valvomoaseman välinen yhteys on toteutettu erillisellä verkkopiuhalla, joka on täytynyt vetää laboratorion lattian halki. Johdotus tällä tavalla on epäsiisti ja voisi aiheuttaa turvallisuusriskin omatoimissa harjoituksissa.

Prosessin toimintaa on demonstroitu tietokoneruudulta, joka on kohtuuttoman pieni opetustilaan nähden. Ajoittain on käytetty myös kannettavaa videotykkiä, mutta sille ei ole erityistä valkokangasta, johon kuvan voisi järkevästi heijastaa.

5.8 Oppimisympäristöön liittyvä dokumentointi

Oppimisympäristöön liittyvä dokumentointi on jäänyt laite- ja ohjelmistopäivitysten yhteydessä osittain puutteelliseksi. Viimeisimmän laitteistouudistuksen jälkeen prosessin I/O-liitäntöihin ja johdotuksiin liittyvä dokumentointi on jäänyt tekemättä. Laitteiston vanhempaa dokumentointia löytyy, mutta se ei kaikilta osin pidä enää paikkaansa. Prosessin ajanmukaisetkin dokumentit ovat suurimmilta osin paperimuodossa, joten dokumentointia on vaikeaa selata ja muokata.

Prosessi- ja automaatiolaitteistoon tulee lähes vuosittain ainakin pientä vikaa. Huolto on suoritettu lähinnä silloin, kun jotain on hajonnut eikä erilaisista vioista ei ole pidetty kirjaa. Tislausprosessiin ja sen automaatiojärjestelmän turvallisuuteen liittyvää dokumentointia on vähäisesti ja se on osaltaan vanhentunutta.

Oppimisympäristön turvallisuuteen liittyvä dokumentointi on lähinnä tislauskolonnin valmistumisen ajalta olevaa etanolin säilytykseen liittyvää dokumentointia.

6. OPPIMISYMPÄRISTÖN PÄIVITTÄMINEN

Tässä luvussa käsitellään, miten luvussa 3 esitettyjä mallipohjaisen systeemisuunnittelun menetelmiä ja SysML-kieltä käytettiin oppimisympäristön ja sen automaatiojärjestelmän päivityksen suunnittelussa. Lisäksi selvitetään mitä hankintoja ja muutoksia tehtiin. Lopuksi esitellään suunnitelma lisämuutoksista, joita oppimisympäristö vaatii tulevia omatoimisia laboratorioharjoituksia sekä tulevaisuuden muuta toimintaa varten.

6.1 Mallipohjainen systeemisuunnittelu oppimisympäristön päivittämisessä

Työn luvussa 3.8 esiteltiin kolme yleisesti käytettyä mallipohjaisen systeemisuunnittelun menetelmää. Näistä RUP-SE on oikeastaan vain kehys, joka tehdään sopivaksi omaan sovelluskohteeseen eikä sitä haluttu tässä työssä ruveta erikseen tekemään, koska sovelluskohde oli ainutkertainen ja menetelmän kehittäminen olisi vaatinut oman aikansa.

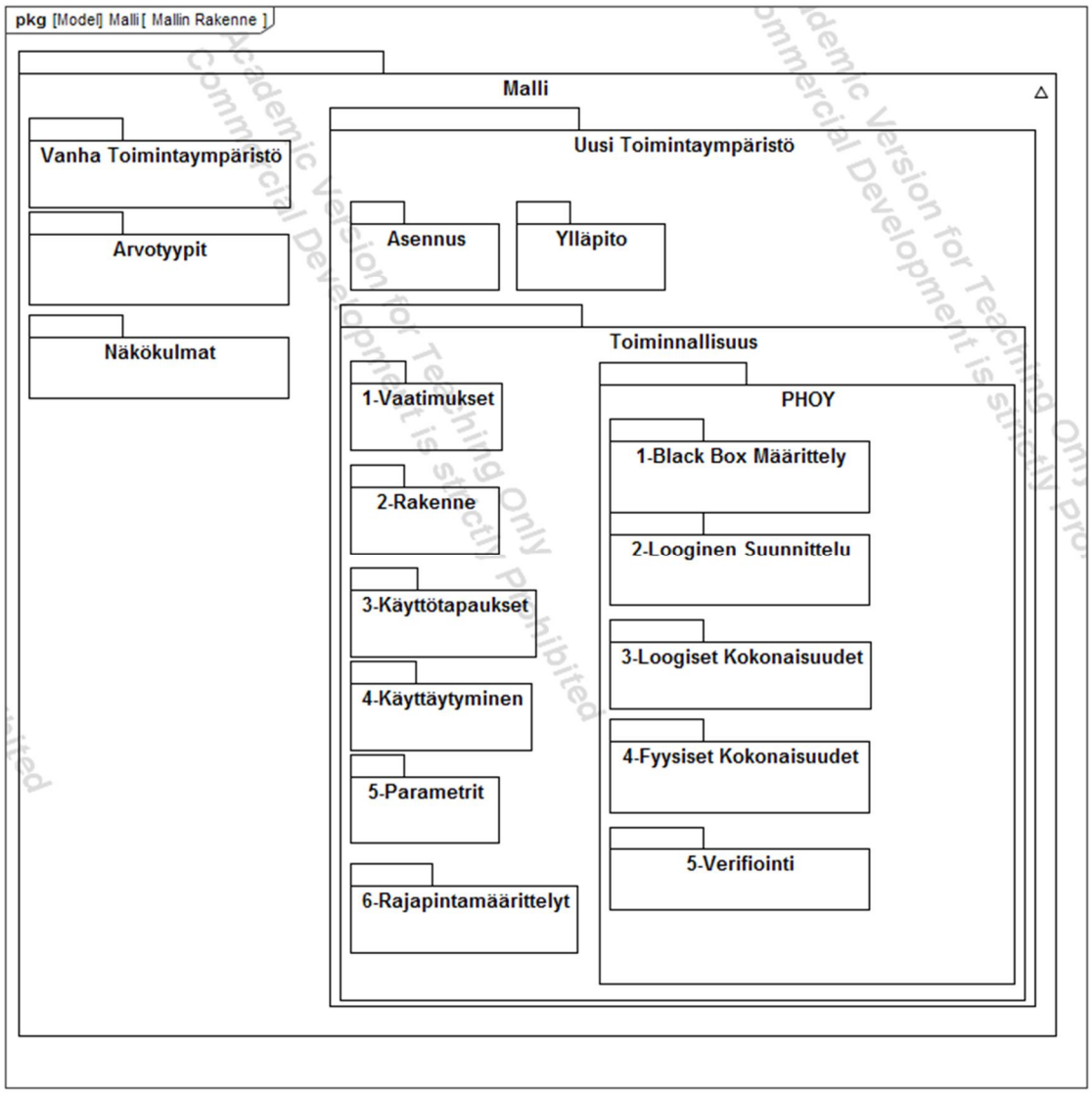
Vertailtaessa SYSMOD ja OOSEM menetelmiä, osoittautui SYSMOD huomattavasti harvemmin käytetyksi ja sen dokumentointi oli OOSEM-menetelmään verrattuna heikompaa. OOSEM-mentelmän käytöstä eri sovelluskohteissa löytyi myös hyviä esimerkkejä [40,70,71]. Menetelmä pitää sisällään sidosryhmien tarpeitten määritysvaiheessa olemassa olevan systeemin karakterisoinnin, joten menetelmä sopii hyvin olemassa olevien systeemien uusimiseen, joka tässä työssä oli kyseessä. Tämä suunnitteluvaihe auttaa tunnistamaan vanhojen systeemien mahdolliset uudelleenkäytettävät osat ja alisysteemit sekä kehitettävät osa-alueet [70]. Oppimisympäristön päivityksen suunnitteluun valittiin edellisten seikkojen perusteella OOSEM-mentelmä.

6.1.1 Mallin laatiminen

Tässä vaiheessa päätettiin mallinnustavoista, siitä mitä systeemin malli sisältää ja kuinka malli organisoidaan. Samalla valittiin myös käytettävä mallinnustyökalu.

Työssä käytettiin systeemin mallin luomisessa samoja nimeämis- ja mallinnuskäytäntöjä kuin lähteen [40] malliesimerkeissä. Mallin elementit nimettiin suomeksi. Mallinnustyökaluksi valittiin Magicdraw UML ja siihen SysML-plugin. Kyseinen ohjelmisto on yksi eniten käytetyimmistä mallipohjaisessa systeemisuunnittelussa ja ohjelmistosuunnittelussa. Myös SysML-kirjallisuuden käsittelemät esimerkkimallit löytyivät valmiina Magicdraw-malleina. [18,40] Lisäksi Systeemitekniikan laitoksella oli valmiiksi akateeminen Magicdraw-lisenssi, eikä sen puolesta tarvinnut tehdä erillisiä hankintoja.

Mallin organisointiin käytettiin OOSEM-mentelmän oletusta mallihierarkiasta, joka on esitetty kuvassa 33. Lisäksi mallinnuksen apuna käytettiin OOSEM SysML-profiilia, joka sisältää eri mallinnusvaiheissa käytettävien mallielementtien stereotyypit.

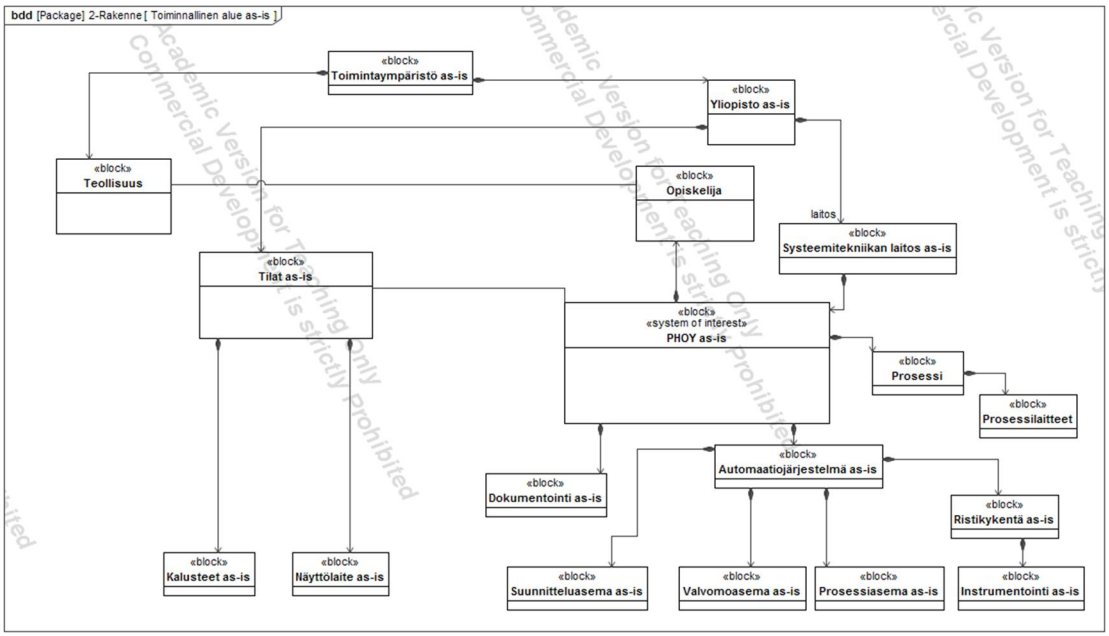


Kuva 33. *Mallirakenne.*

6.1.2 Sidosryhmätarpeiden analysointi

Tässä vaiheessa karakterisoitiin aluksi olemassa oleva systeemi ja muodostettiin vanhan systeemin toimialamalli, joka on esitetty kuvassa 34. Karakterisointiin käytettiin tislausprosessiin ja oppimisympäristöön liittyvää dokumentointia sekä itse oppimisympäristön fyysistä havainnointia. Vanhan systeemin mallin perusteella tehtiin havainnot puutteista ja parannuskohteista yhteistyössä sidosryhmien kanssa. Kokonaisuudet joihin muutoksia ajateltiin todennäköisesti tehtävän, sisältää nimessä *as-is*-liitteen. Tämä selventää lohkon kuvaavan sen tämänhetkistä tilaa. Varsinaisen kehittämisen kohteena olevan oppimisympäristön lohkoon on käytetty *<<system of interest>>*-stereotyyppiä

määrittämään sen olevan tutkimuksen kohteena. Kehittämisen kohteena systeemistä on käytetty mallissa lyhennettä PHOY, jolla tarkoitetaan prosessien hallinnan oppimisympäristöä.



Kuva 34. *Oppimisympäristöön liittyvä toimialamalli.*

Sidosryhmien tarpeita selvitettiin mahdollisimman monesta lähteestä, jotka liittyivät kehitettävään oppimisympäristöön. Tärkeimpinä lähteinä toimivat nykyiset sekä tulevat automaation ja prosessien hallintaan liittyvien kurssien laboratorioharjoitusten harjoitusmonisteet, opetushenkilöstö ja opiskelijat.

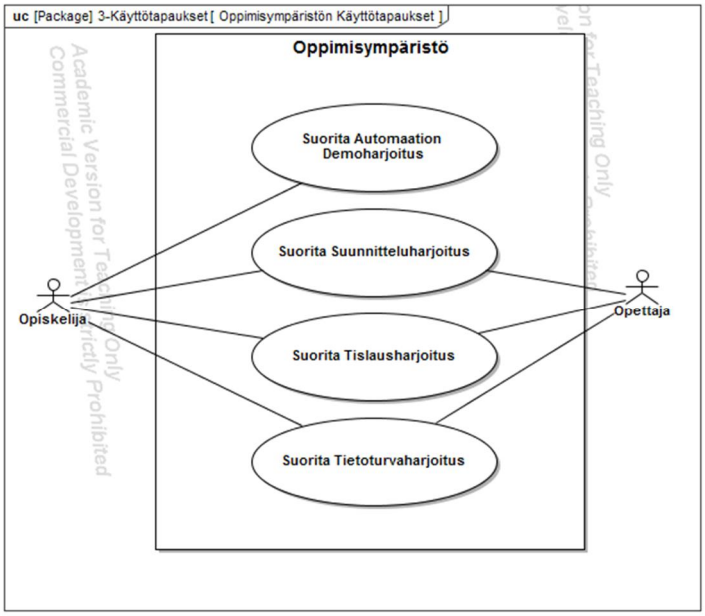
Opetushenkilöstöä haastateltiin lähinnä ryhmätapaamisissa, missä keskusteltiin tarpeista oppimisympäristöä kohtaan sekä käytiin läpi erilaisia vaihtoehtoja. Opiskelijoiden toiveita oppimisympäristöä kohtaa kartoitettiin kyselylomakkeilla laboratorioharjoitusten yhteydessä erillisessä tutkimuksessa laboratoriotilojen käyttöön liittyen [72]. Myös tämän oppimisympäristön uudistusprojektin kanssa samaan aikaan tehty oppimisympäristön turvallisuuteen liittyvä diplomityön materiaali toimi vaatimusten lähteenä. Turvallisuutta selvittävän työn myötä selvisi alkoholilainsäädäntöön, työturvallisuuteen sekä oppimisympäristössä opetukseen liittyviä vaatimuksia. [74]

Ei-toiminnalliset sidosryhmävaatimukset mallinnettiin sanallisesti vaatimuskaavioiden avulla. Vaatimukset kuvattiin *Extended Requirement*-mallielementin avulla. Kyseessä on normaalista vaatimuselementistä johdettu stereotyyppi, joka mahdollistaa ylimääräisten ominaisuuksien kuvaamisen sanallisille vaatimuksille. Tässä tapauksessa tarpeelliseksi tuli vaatimuksen lähteen määrittämien, jotta lopullisen toteutuksen jäljitettävyys voidaan johtaa tiettyyn sidosryhmään. Tässä vaiheessa määritetyt vaatimukset olivat yleisessä muodossa ja määrittelivät, mitä systeemin haluttiin pääpiirteittäin toteuttavan. Sanalliset sidosryhmävaatimukset on kuvattu taulukkomuodossa kuvassa 35.

#	Name	Text	Source
1	Mittaukset	Prosessin virtasignaaleita tulisi pystyä mittaamaan	ASE-1130 Opetus
2	Realistisuus	Oppimisympäristön tulisi vastata autenttista automaatioympäristöä	ASE-1130 Opetus,
3	Opetusmateriaali	Opetusmateriaalin pysyttävä mahdollisimman ennallaan	ASE-1130 Opetus, ASE-4030 Opetus
4	Dokumentointi	Oppimisympäristöön liittyvä dokumentointi ajantasaista ja monipuolista	ASE-1130 Opetus, Ylläpito
5	Kalusteet	Paremmat tuolit ja kirjoitustasot	ASE-1130 Opiskelijat, ASE-2170 Opiskelijat, ASE-4030 Opiskelijat
6	Tilat	Työskentelytilan tulee olla siisti ja avara	ASE-1130 Opiskelijat, ASE-2170 Opiskelijat, ASE-4030 Opiskelijat, ASE-4030 Opetus, Ylläpito
7	Melun vähentäminen	Melua täytyisi pystyä vähentämään opetustilanteessa	ASE-1130 Opiskelijat, ASE-4030 Opetus
8	Laitteisto	Laitteiston tulee olla modernia ja monipuolista	ASE-1130 Opiskelijat, ASE-4030 Opiskelijat, ASE-7610 Opetus
9	Prosessidatan tallennus	Prosessidataa pystyttävä tallentamaan	ASE-4030 Opetus, ASE-7610 Opetus
10	Prosessin toiminnan havainnointi	Prosessin valvontaan hankittava suurempi näyttö	ASE-4030 Opiskelijat, ASE-4030 Opetus
11	Toimintavarmuus	Prosessin toimintavarmuutta parannettava	ASE-4030 Opiskelijat, ASE-4030 Opetus
12	Prosessin nopeus	Prosessin ylösajoa ja vasteita pyrittävä nopeuttamaan	ASE-4030 Opiskelijat, ASE-4030 Opetus, ASE-7610 Opetus
13	Verkko- ja väyläliikenne	Verkko- ja väyläliikenteen tulee olla mahdollisimman monipuolista	ASE-7610 Opetus

Kuva 35. *Sanalliset sidosryhmävaatimukset.*

Sidosryhmien vaatimusten perusteella määritettiin tehtävävaatimukset, jotka kuvaavat systeemiltä vaadittua toiminnallisuutta. Tehtävävaatimukset täyttävät oppimisympäristöön liittyvät laboratorioharjoitusten toiminnalliset vaatimukset mallinnettiin käyttötapausten muodossa, jotka ovat esitettynä kuvassa 36. Käyttötapausten suorittamista mallinnetaan tarkemmin seuraavissa suunnitteluvaiheissa. Suunniteltavan systeemin täytyy tukea näitä käyttötapauksia.

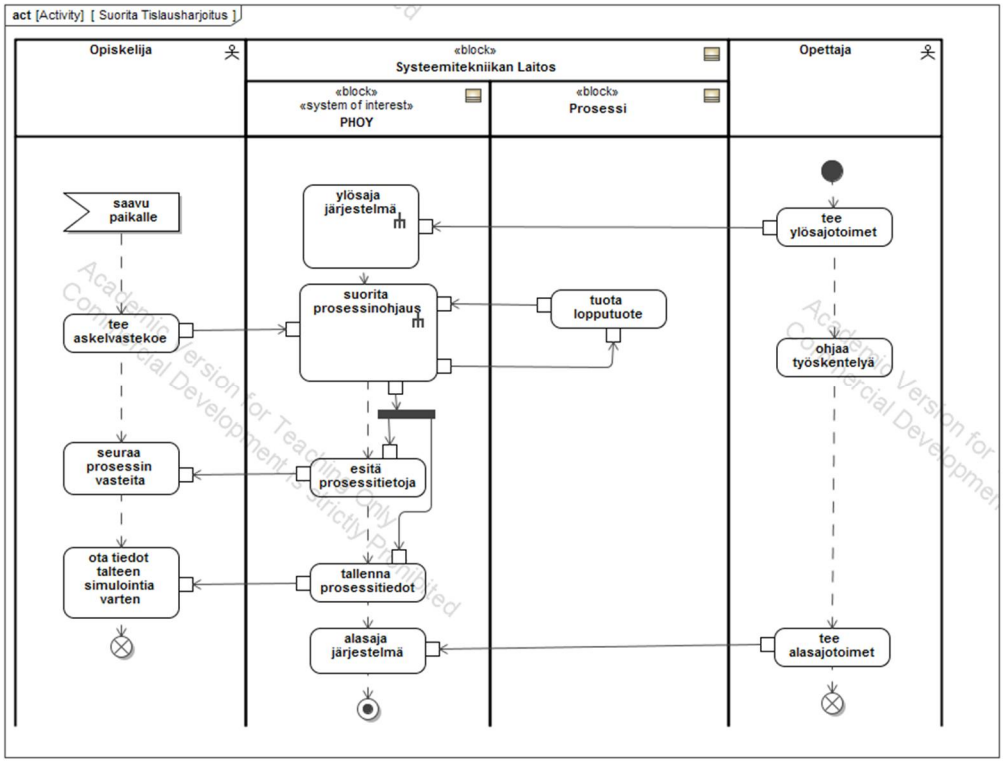


Kuva 36. *Oppimisympäristöön liittyvät käyttötapaukset.*

6.1.3 Systeemivaatimusten analysointi

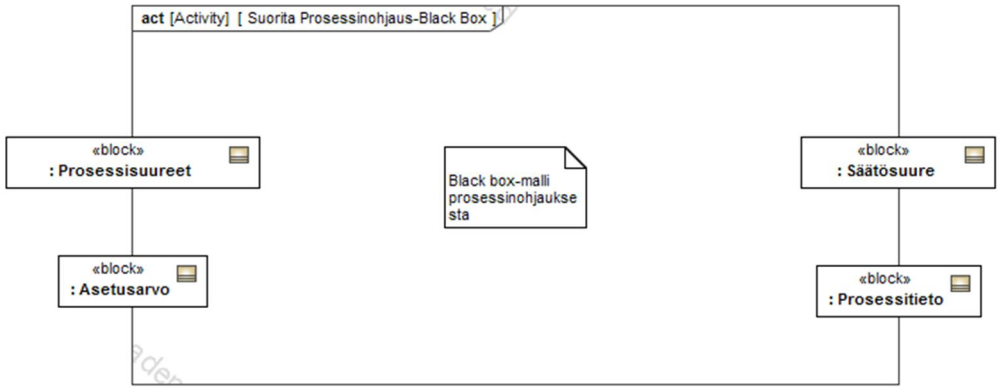
Tässä vaiheessa määritettiin oppimisympäristön vaatimukset black box-mallina. Systeemivaatimukset muodostuvat sidosryhmätarpeiden vaiheessa määritettyjen käyttötapausten skenaarioanalyysien perusteella. Analysointi tapahtuu muodostamalla käyttötapauksia kuvaavat aktiviteettikaaviot. Yhtä käyttötapausta voi vastata yksi tai useampi

kaavio. Tislausharjoitusta kuvaava aktiviteettikaavio on esitetty kuvassa 37. Suunniteltavan systeemin tulee mahdollistaa harjoitusta kuvaavan aktiviteettikaavion sisältämät toiminnot.



Kuva 37. *Tislausharjoitusta kuvaava aktiviteettikaavio.*

Kuvan 37 skenaarion toiminnot mallinnettiin yksinkertaisilla aktiviteettikaavioilla, jotka kuvaavat systeemiltä vaadittavien toimintojen inputit ja outputit. Prosessinohjausta kuvaava kaavio black box-mallina on esitetty kuvassa 38.



Kuva 38. *Prosessinohjauksen toimintaa kuvaava aktiviteettikaavio.*

Lisäksi määriteltiin rajapinnat, joiden kautta systeemiin vaikutetaan. Rajapintoja kuvaavien porttityyppien nimiin on lisätty lyhenne IF (*interface*). Systeemille on lisäksi määritetty tekniset mittarit, jotka mittaavat systeemin tärkeää suoriutumista omistavalle ta-

holle eli systeemitekniikan laitokselle. Motivaatiota kuvaavaa teknistä mittaria voidaan mitata esimerkiksi kurssipalautteen avulla numeroina. Viiteominaisuutena systeemille on kuvattu <<*store*>>-stereotyypillä, että tarvitaan tietojen tallennusmahdollisuus. Kehitettävää systeemiä kuvaava black box-määrittely on kuvattu lohkona, joka on esitetty kuvassa 39.

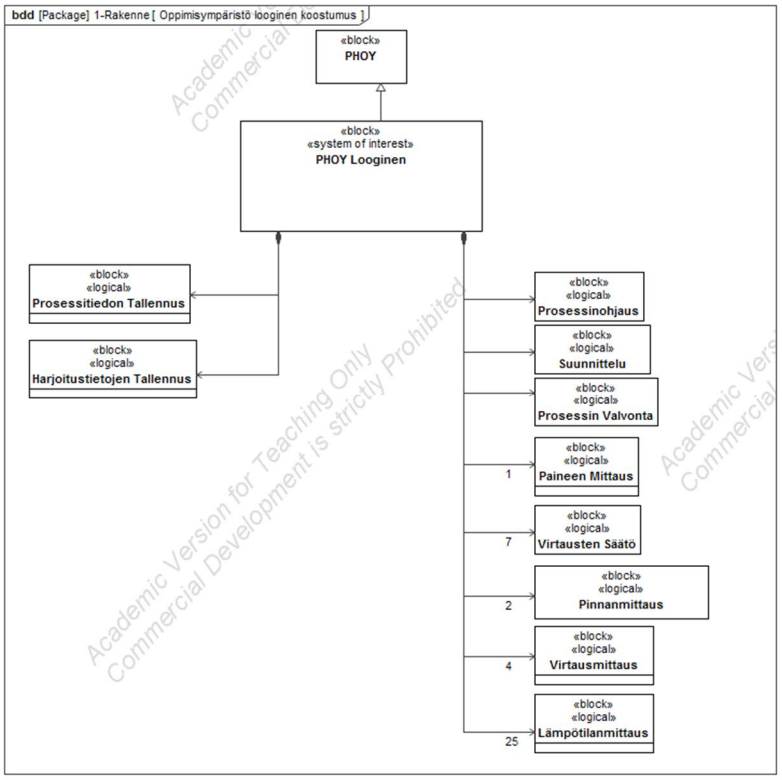
Kuva 39. *Systeemin black box-mallin määrittävä lohko.*

Systeemivaatimusten analysointivaiheessa määritettiin myös suunnittelun rajoitukset. Oppimisympäristöön liittyvää tislausprosessia ei muuteta, joten sitä ei mallinneta systeemin mallissa kuin rajapinnan muodossa prosessinohjaukseen. Automaatiojärjestelmän laitekaappia ei uusita ja sen sisältämä vanha sähkönsyöttö pysyy ennallaan. Laitekaapin ennallaan pitäminen aiheutti rajoituksia esimerkiksi kaappiin asennettavien laitteiden dimensioihin. Oppimisympäristössä käytetyn automaatiojärjestelmän automaatiosovellus ja ohjelmistojen toimivuus haluttiin säilyttää mahdollisimman alkuperäisen kaltaisina. Tämän vuoksi rajoituksena asetettiin Metson prosessiasemien käyttö sekä Metson valvomo- ja suunnitteluohjelmistojen käyttö, jotta käytetyt valmiit sovellukset voidaan säilyttää ja opetusmateriaaliin ei tarvitse tehdä suuria muutoksia. Lisäksi ohjelmistot haluttiin virtualisoida olemassa olevalle systeemitekniikan laitoksen virtuaaliklusterille.

6.1.4 Loogisen arkkitehtuurin määritys

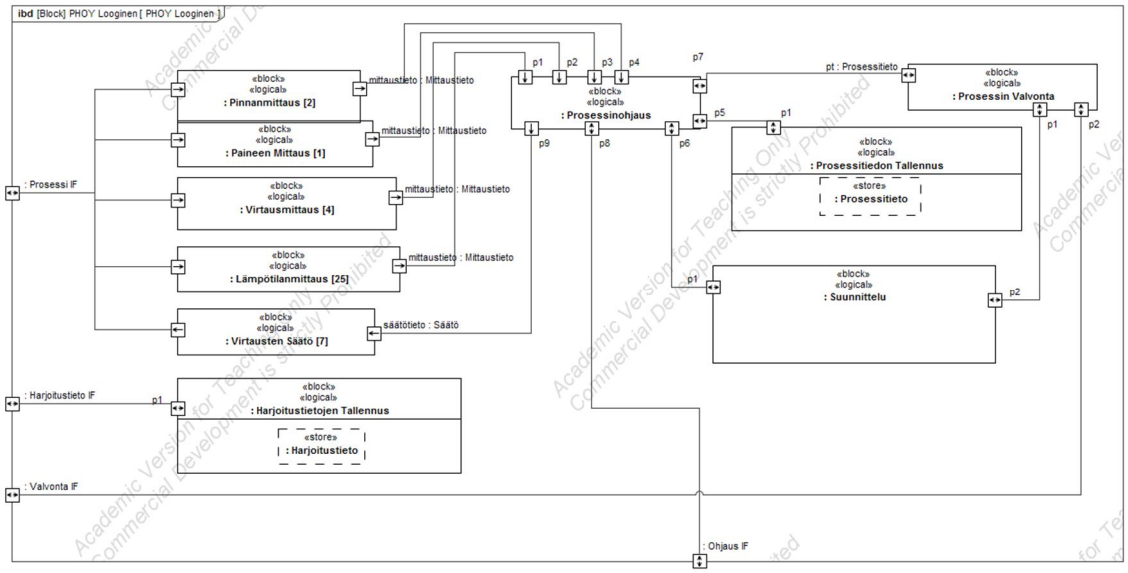
Tässä mallinnusvaiheessa black box-mallin sisältävä toiminnallisuus mallinnettiin loogisista osista ja niiden välisistä suhteista. Loogista systeemiä kuvaava lohko on yleistys black box-mallin lohkosta ja sisältää siis kaikki siihen kuuluvat ominaisuudet. Loogiset osat kuvataan lohkon määrittelykaaviossa, joka on esitetty kuvassa 40. Lohkoissa käytetään OOSEM-profiilin <<*logical*>>-stereotyyppiä selventämään niiden olevan pelkäs-

tään loogisia osia. Tässä vaiheessa ei oteta siihen kantaa mitkä osat koostuvat laitteistosta ja mitkä ohjelmistoista, eikä siitä kuinka osat systeemiin sijoitetaan. Prosessi itsessään ei muutu ja säätöohjelmisto säilyy ennallaan. Mittausten ja lähtöjen määrä pysyy samana ja ne ovat merkitty kerrannaisina loogisten osien yhteydessä.



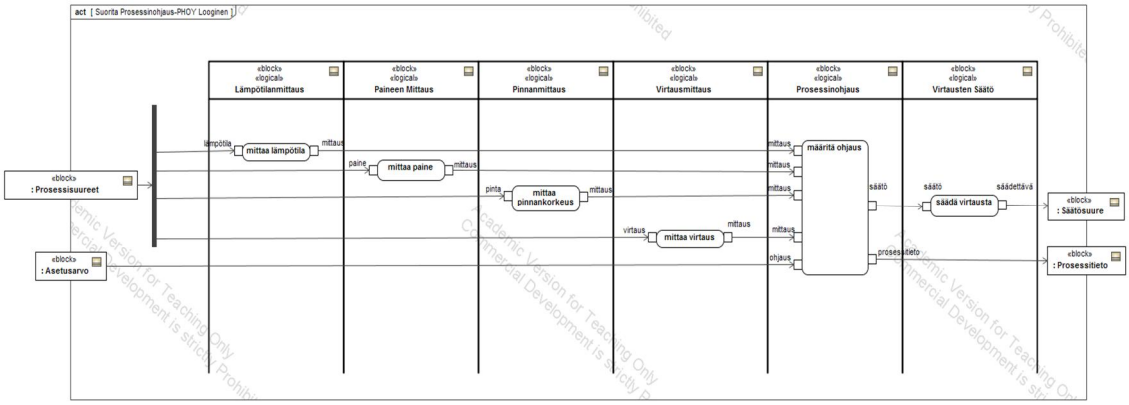
Kuva 40. *Systeemin mallin loogiset osat.*

Loogiset osat sisältävästä lohkon määrittelykaaviosta tehtiin myös sisäinen lohkokaavio, joka mallintaa loogisten osien välisiä suhteita systeemissä. Loogisen arkkitehtuurin sisäinen lohkokaavio on esitetty kuvassa 41.



Kuva 41. *Loogisten osien sisäinen lohkokaavio.*

Lisäksi muodostettiin aktiviteettikaavio, joka kuvaa kuinka loogiset osat toteuttavat systeemin vaaditut toiminnallisuudet. Aktiviteettikaaviot kuvaavat loogisten osien toimintaa systeemin vaatimuksia vastaten. Prosessinohjausta kuvaava aktiviteettikaavio, jossa toiminnallisuus on jaettu loogisten osien kesken, on esitetty kuvassa 42.



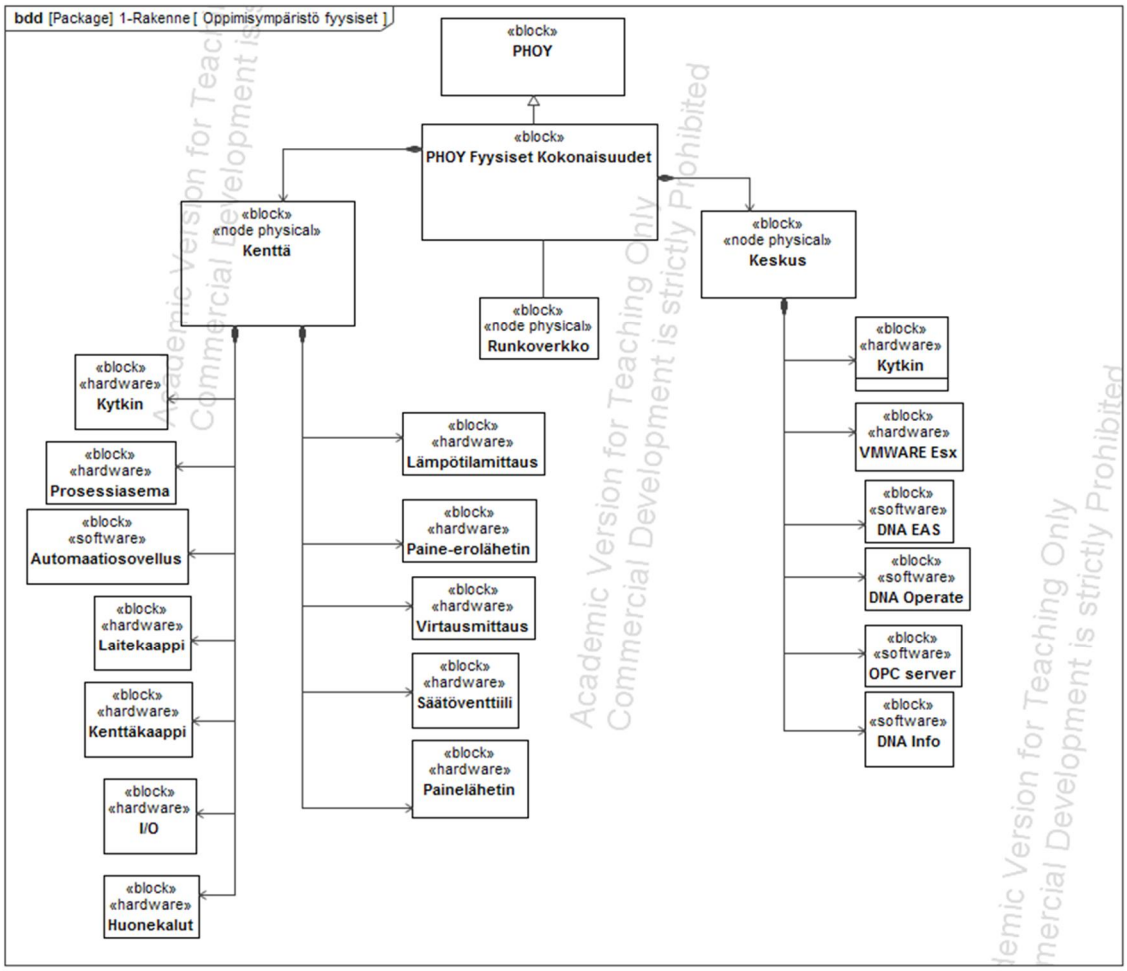
Kuva 42. *Prosessinohjauksen loogista toteutusta kuvaava aktiviteettikaavio.*

Systeemin malliin muodostettiin tässä vaiheessa looginen arkkitehtuuri, kuinka systeemivaatimuksissa selvitetyt toiminnallisuudet systeemissä kannattaa toteuttaa. Myös systeemin loogisten kokonaisuuksien rajapinnat määritettiin malliin tässä vaiheessa.

6.1.5 Fyysisen toteutuksen valinta ja elinkaaren tukimallit

Edellisessä vaiheessa määritetyt loogiset osat ryhmiteltiin kokonaisuuksiin sen mukaan miten niiden sisältämä tullaan sijoittamaan systeemissä. Sen jälkeen jaettiin niiden toi-

minta fyysisten osien kesken. Kokonaisuudet ovat kuvattu mallissa OOSEM-kirjaston <<*node physical*>>-stereotyyppiä käyttäen. Fyysisillä osilla tarkoitetaan konkreettista laitteistoa ja ohjelmistoa systeemissä. Osat eriteltiin kokonaisuuksiin kenttä ja keskus. Lisäksi mallissa on viiteassosiaatio runkoverkkoon, joka toimii kokonaisuuksien yhdistäjänä. Kenttäosassa ovat osakomponentit, jotka liittyvät ja asennetaan itse fyysisen oppimisympäristön tiloihin. Keskusosassa ovat systeemin osakomponentit, jotka liittyvät virtualisoituun ohjelmistokokonaisuuteen, joka oli suunnittelurajoituksena systeemivaatimuksissa. Laitteet tai oikeasti fyysisesti systeemissä esiintyvät komponentit ovat esitetty <<*hardware*>>-stereotyyppinä. Ohjelmistot ovat esitetty <<*software*>>-stereotyyppiä käyttäen. Systeemin fyysiset osat ovat mallinnettu lohkoina kuvassa 43. Useat osista kuten esimerkiksi ohjelmistot ovat tarkasti määritettyjä systeemivaatimuksissa esitettyjen suunnittelurajoitusten takia.

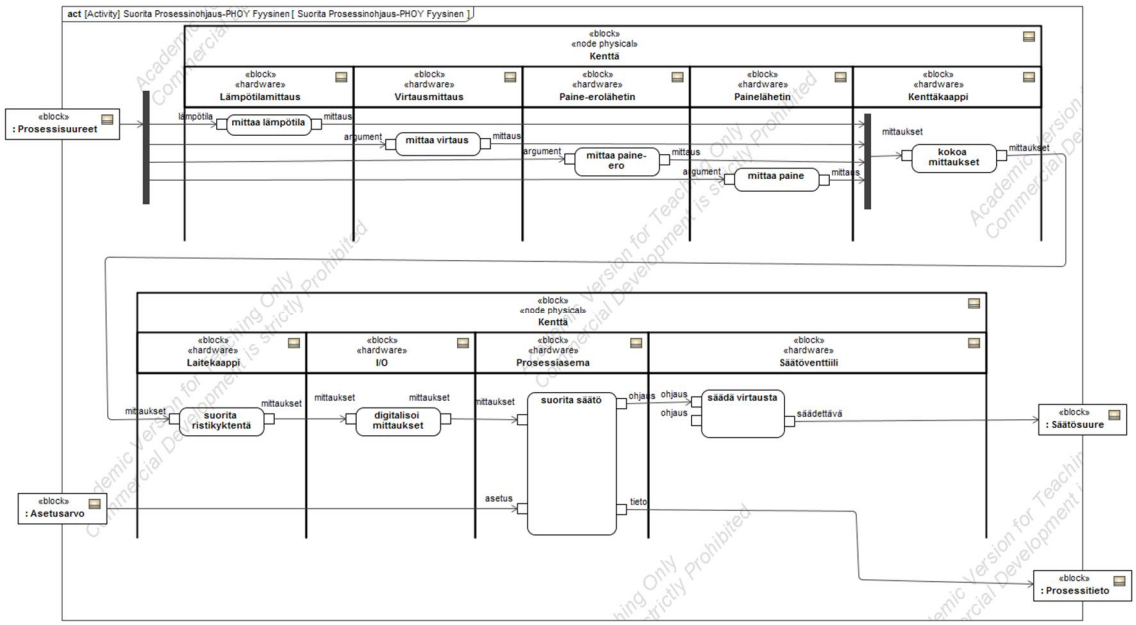


Kuva 43. *Systeemin fyysiset osat kokonaisuuksiin eriteltynä.*

Fyysisten osien keskinäisiä suhteita mallinnettiin lisäksi sisäisen lohkokaavion avulla kuten loogisten osien tapauksessa kuvan 40 esimerkissä.

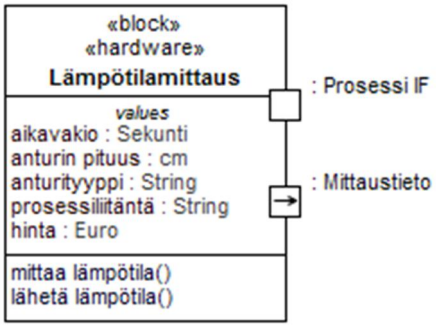
Edellisten vaiheiden systeemin toiminnallisuutta kuvaavien kaavioiden pohjalta tehtiin toiminnallisuutta kuvaavat kaaviot, joissa toiminnallisuus on jaettu fyysisten osien kes-

ken. Kuvassa 44 on esitetty edellisissäkin vaiheissa mallinnettu prosessinohjausta kuvaava aktiviteettikaavio.



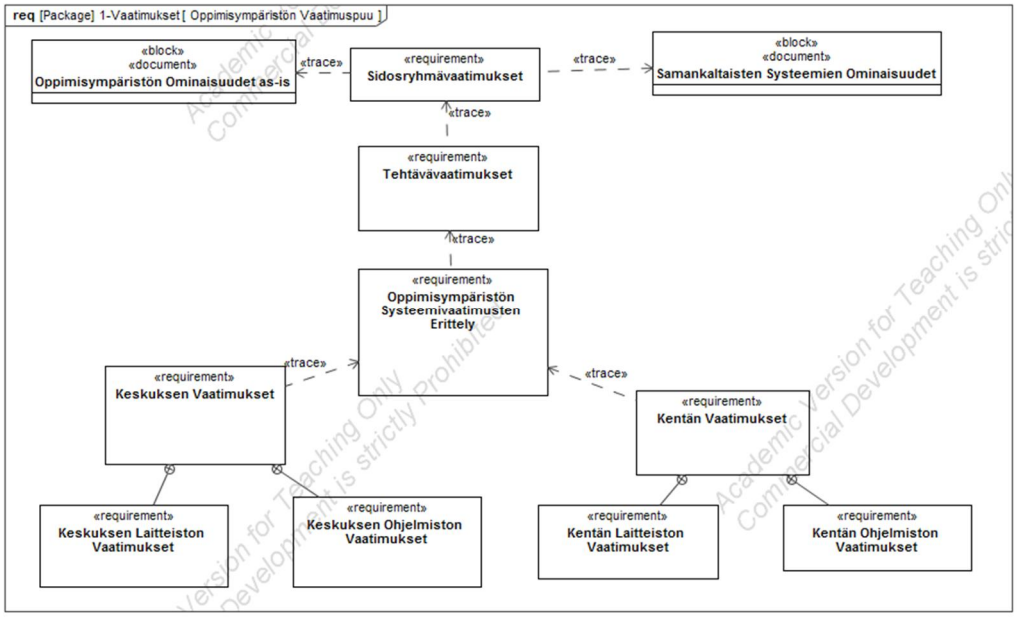
Kuva 44. *Fyysisten osien kesken jaettu ohjausta kuvaava aktiviteettikaavio.*

Fyysisille osille määritettiin tietyt ominaisuudet, joille voitiin asettaa vaatimuksia. Näiden vaatimusten perusteella voitiin tehdä valinta tarkasta osan toteutuksesta tai hankinnasta. Fyysisten osien ominaisuuksien mallintamisesta on esimerkki lämpötilamittauksen ominaisuuksista kuvassa 45.



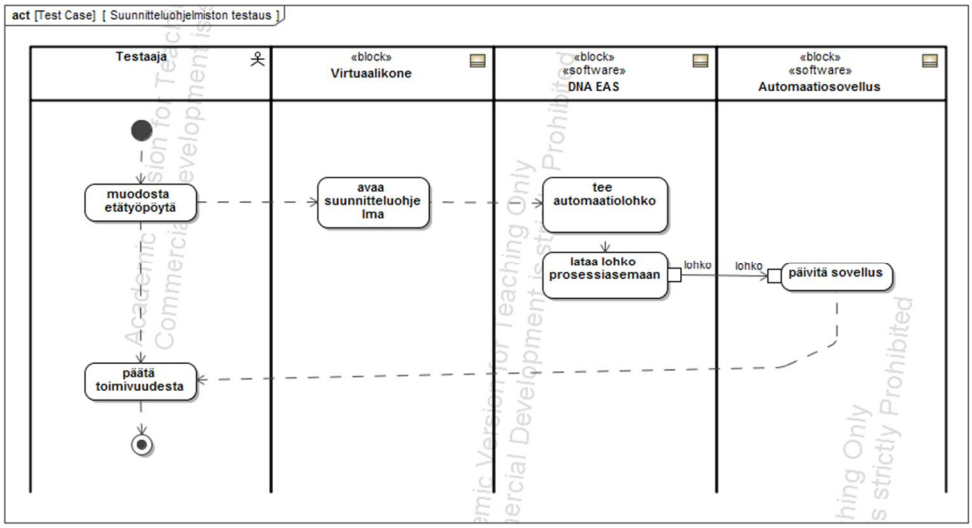
Kuva 45. *Fyysisten osien ominaisuuksien mallintaminen.*

Lopullisessa systeemin mallissa pidettiin huolta jäljitettävyydestä alkuperäisten sidosryhmätarpeiden ja systeemin osakomponenttien välillä, jotta voidaan varmistaa, että alkuperäiset tarpeet toteutuvat. Kuvassa 46 on esitetty pääpiirteinen puumuotoinen vaatimusten jäljitettävyyttä esittävä kaavio sidosryhmätarpeista erilaisiin osavaatimuksiin. Oppimisympäristön ominaisuudet-dokumentti kuvaa vanhaan oppimisympäristöön liittyvää dokumentointia. Samankaltaisten systeemien ominaisuudet-dokumentti mallintaa työssä tehtyä tutkimusta samankaltaisten ympäristöjen ominaisuuksista sekä automaatiojärjestelmistä.



Kuva 46. *Systeemin vaatimusten jäljitettävyys.*

Lopullisen valinnan jälkeen systeemin testausta varten muodostettiin systeemin mallin pohjalta testitapaukset aktiviteettikaavioiden muodossa. Testitapaukset määriteltiin instrumentaation testaukseen, automaatiojärjestelmän testaukseen sekä erilaisia vikatilanteita varten. Suunnitteluohjelmiston testitapauksen aktiviteettikaavio on esitetty esimerkkinä mallinnetusta testitapauksesta kuvassa 47.



Kuva 47. *Suunnitteluohjelmiston testitapaus.*

Systeemin asennusta ja ylläpitoa varten tehtiin systeemin elinkaaren toiminnan mahdollistavat tukimallit. Niissä on määritetty pääasiassa vaatimukset systeemin asennusta ja ylläpitoa varten. Lisäksi on määritetty tarvittava dokumentointi jatkoa ajatellen.

Erilaisia toteutusvaihtoehtoja vertailtiin edellisissä vaiheissa määritettyihin vaatimuksiin sekä määritettiin järkevintä kokoonpanoa hankintabudjetin rajoissa. Tarkat hankinnat on

selostettu työn seuraavissa luvuissa. Lisäksi kerrotaan mitä muutoksia itse oppimisympäristön suhteen täytyy tehdä tulevaisuudessa, jotta alkuperäiset sidosryhmien tarpeet täyttyvät ja omatoimiset laboratorioharjoitukset mahdollistuvat.

6.2 Automaatiojärjestelmän ja laitteiston päivitys

Edellisten vaiheiden ja sidosryhmätarpeiden perusteella havaittiin, että oppimisympäristön automaatiojärjestelmä oli keskeisessä osassa lopullista fyysistä ratkaisua valittaessa. Prosessiin liittyvän automaatiojärjestelmän laitteet sekä toteutustapa olivat vanhentuneita. Automaation tietoliikenteeseen liittyvien kurssien vastuuhenkilöt toivat esiin tarpeen uusille väyläratkaisuille automaatiojärjestelmässä. Opiskelijoiden palautteesta tuli ilmi toiveet modernimmalle automaatiolaitteistolle [72]. Myös järjestelmän hajauttaminen nähtiin tarpeelliseksi.

Ennen uudistusta prosessinohjauksessa oli käytetty Metso DNA automaatiojärjestelmää. Metso DNA on yleisesti käytetty automaatiojärjestelmä prosessiteollisuudessa ja erilaisissa automaatioratkaisuissa. Tislauskolonnin ohjaussovellus sekä valvomokuvat ovat tehty kyseiselle järjestelmälle ja sovelluksen siirto päivitettyyn versioon onnistuu helposti. Laboratoriossa opettava laitoksen henkilökunta on koulutettu Metso DNA järjestelmän käyttöön ja opetusmateriaali pohjautuu kyseiseen järjestelmän toimintaan. Koko järjestelmän uusiminen olisi aiheuttanut suurten kustannusten lisäksi mahdollisia katkoksia opetukseen. Rajoitus Metson prosessiasemien käytöstä määritettiin systeemivaatimuksissa. Aikataulun toteutuminen sekä vähintään saman toiminnallisuuden säilyttäminen täyttävät alkuperäiset sidosryhmävaatimukset.

Automaatiojärjestelmältä vaadittiin lisäksi muiden prosessien liittämisen mahdollisuus. Projektin aikana tarpeelliseksi katsottiin paperikoneen perälaatikkoa mallintavan laboratorioharjoituslaitteiston liittäminen tislauskolonnin pääautomaatiojärjestelmään.

Projektin vaatimuksiksi johdettiin oppilaitten ja opetushenkilökunnan puolesta tulleet toiveet modernimmasta instrumentoinnista. Tislausprosessin instrumentointiin haluttiin tuoda moderneja ratkaisuja sekä monipuolisuutta opetuskäyttöä ajatellen. Myös prosessin toimintavarmuutta haluttiin parantaa uusimalla vanhaa instrumentointia. Lisäksi oli toivomuksena nopeuttaa prosessin ylösajoa sekä mahdollisesti prosessivasteita mahdollistamalla syöttövirtauksen esilämmityksen käyttäminen.

Fyysisen arkkitehtuurin osavaatimusten perusteella toimittajalta pyydettiin tarjous mahdollisista vaihtoehdoista prosessiasemiksi, ohjelmistoiksi sekä kenttälaitteiksi. Toimittajan ehdottamista laitteista muodostettiin sopiva hankintakokonaisuus käytettävän budjetin ja vaatimusten perusteella.

Laitteiden valinnassa täytyi lisäksi harkita minkälaiseen tilaan kyseiset laitteet tulevat ja vaaditaanko niiltä esimerkiksi räjähdysvaarallisiin tiloihin soveltuvuutta. Tislausproses-

silaboratorion turvallisuutta käsittelevässä diplomityössä selvisi, että tiloille on tehty TTY:n puolesta riskiarviointi ulkopuolisen yrityksen toimesta [74]. Tämän selvityksen mukaan tislauskolonnia voidaan pitää suljettuna systeeminä, eikä tilaa tarvitse pitää räjähdysvaarallisena.

Automaatiojärjestelmän olemassa oleva laitekaappi päätettiin säilyttää. Kaapin sisältämä vanha sähkönsyöttö voitiin säilyttää sellaisenaan. Kaapista poistettiin vanha Metson sulautettu I/O ja sen tilalle päätettiin hankkia modernimpi ratkaisu. Kaapista poistettiin lisäksi vanha tutkimuskäytössä ollut Omronin logiikka tarpeettomana. Vanhan I/O:n tilalle hankittiin nykyaikaisempi Metso ACN MIO M80 ratkaisu. I/O koostuu kehikosta, tehonsyöttöyksiköstä, väyläohjaimesta sekä yhdeksästä I/O-kortista. Kehikossa on tilaa yhteensä 16 kortille, joten järjestelmän laajentamiene on tulevaisuudessa mahdollista. Analogiatulokortteja on kuusi, joista yksi kykenee HART-kommunikointiin. Analogialähtökortteja on kaksi, jotka molemmat ovat HART-kommunikointikortteja. Lisäksi I/O sisältää digitaaliset tulo- ja lähtökortit.

Vanha PC-pohjainen prosessinohjaus poistettiin käytöstä. Tislausprosessin ohjaus päätettiin hajauttaa kahdelle eri prosessiasemalle. Molempiin prosessiasemiin vaadittiin tuki Modbus-protokollalle. Prosessitilan yläkertaan vanhaan laitekaappiin sijoitettiin Metson ACN C20 prosessiasema. Kyseinen asema toimii järjestelmän pääprosessiasemana. Prosessin alakertaan hankittiin erillinen Metson MR prosessiasema, joka liittyy pääprosessiasemaan Ethernetin välityksellä. Alakerran prosessiasemaan liitetään järjestelmän kenttäväylälaitteet. Alakertaan sijoitettavalle prosessiasemalle hankittiin lisäksi kotelo.

Hajautettavuus toteuttaa alkuperäisen sidosryhmävaatimuksen. Lisäksi eri prosessiasemien välisen liikenteen tutkimus oli yhtenä vaatimuksena. Tislausprosessin eri prosessiasemien sekä myöhemmin toteutettava langaton yhteys perälaatikkoprosessista täyttävät sen vaatimuksen.

Paperikoneen perälaatikkoa demostroivaa prosessia varten hankittiin Metson ACN SR1 prosessiasema sekä tarvittavat I/O-kortit ja kotelointi. Perälaatikon prosessiasema liitetään pääprosessiasemaan langattoman verkon välityksellä. Lisäksi perälaatikkoa mallintavaa demoa varten tehtiin tarvittava automaatio-ohjelmisto toimittajan puolesta. Perälaatikkolaboratorioharjoituksia voidaan tulevaisuudessa ajaa esimerkiksi kannettavan tietokoneen tai tabletin avulla.

Metso DNA-automaatiojärjestelmän ohjelmistot virtualisoitiin Systeemitekniikan laitoksen VMWare ESX virtuaalikoneelle. Virtuaaliseksi toteutettiin neljä eri palvelinta. Yhdellä palvelimella toimii DNA Operate eli varsinainen valvomosovellus, yhdellä OPC-palvelin, yksi toimii EAS-suunnittelupalvelimena. Valvomo- ja suunnitteluohjelmistot päivitettiin uudempiin versioihin. Neljännelle virtuaalipalvelimelle toteutetaan uutena ominaisuutena järjestelmään Metso DNA info-palvelu, jonka avulla voidaan

tallentaa prosessitietoa tietokantaan ja lukea sitä myöhemmin. Palvelu nähtiin tarpeelliseksi tulevassa tutkimus- ja opetuskäytössä.

Virtualisoinnin avulla prosessin valvominen ja suunnittelutehtävien suorittaminen voidaan tehdä etätyöpöytäyhteydellä. Tämä mahdollistaa tislausprosessiin liittyvien automaatiosuunnitteluharjoitusten tekemisen esimerkiksi mikroluokassa. Kuitenkin varsinaiset prosessikokeet voidaan edelleen suorittaa kolonnilaboratoriossa kannettavan tietokoneen tai tabletin avulla. Näin ollen kolonnilaboratoriossa ei tarvitse säilyttää ollenkaan kiinteätä tietokonelaitteistoa prosessiasemia lukuun ottamatta. Tämä osaltaan vähentää turhien resurssien käyttämistä.

Virtualisoitu järjestelmä tarjoaa myös aiempaa laajemmat mahdollisuudet itse prosessin sovelluksen muokkaukseen opetuskäytössä. Virtuaalikoneesta voidaan ottaa tallenne koneen sen hetkisestä tilasta ja palauttaa se samaan tilaan myöhemmin. Mahdolliset suunnitteluvirheet voidaan kumota tämän avulla. Virtualisoinnin yhteydessä tislausprosessin automaatiosovelluksen valvomokuvat muokattiin ajanmukaisiksi.

Järjestelmän kenttäkaapelointia uusittiin. Osa virtaviestikaapeloinnista johdotettiin alakertaan sijoitettavaan erilliseen kenttäkaappiin. Kenttäkaapilta vedettiin runkokaapeli varsinaiselle I/O–kaapille, jossa ristikytkentä tapahtuu. Tällä tavalla osa prosessin kaapeloinnista toteutettiin samalla lailla kuin se usein teollisuuskohteissa on tehty. Lisäksi kenttäkotelon varustettiin katkaistavilla riviliittimillä, joka mahdollistaa signaalimittausten tekemisen johtimista.

Vanhan järjestelmän ristikytkennältä liitos I/O-korteille oli aikaisemmin toteutettu lattakaapeleilla, jotka olivat edellisen järjestelmän päivityksen aikana purettu osikseen ja johdotettu I/O-korteille epästandardilla tavalla. Nyt laitekaapin ristikytkentä toteutettiin järjestelmällisesti ja se dokumentoitiin huolellisesti. Myös ristikytkennässä on katkaistavat riviliittimet, joten signaalien mittaus ja simulointi onnistuu sen kautta. Uusi toteutustapa on havainnollinen opetusta ajatellen.

Uutena automaation tietoliikenneratkaisuna hankittiin järjestelmään WirelessHART-gateway ja yhteen kenttälaitteeseen liitettävä WirelessHART-adapteri. Adapteri kommunikoi gatewayn kanssa HART-protokollaa käyttäen. Gateway liittyy prosessinohjaukseen ethernetin kautta. Erilaisten tietoliikenneratkaisujen myötä haluttiin tuoda monipuolisuutta opetus ja tutkimuskäyttöön, joka oli yhtenä sidosryhmävaatimuksena.

Tislausprosessin automaatiojärjestelmä oli aiemmin liitetty suoraan Internetiin ja sillä oli oma IP-osoitteensa. Tämäntyyppinen ratkaisu ei ole tietoturvallinen. Tavallisesti teollisuusprosessissa automaatiojärjestelmä toimii omassa verkossaan, joka on liitetty palomuurin kautta toimistoverkkoon ja uuden palomuurin kautta Internetiin.

Tislauskolonnin uusi automaatiojärjestelmä liitettiin Systeemitekniikan laitoksen vikasietoiseen automaation runkoverkkoon. Automaation runkoverkko ja kolonnin auto-

maatiojärjestelmän uudistus ovat osa Tampereen teknillisen yliopiston TUTCyberLabs-hanketta. Runkoverkkoon liitetään myös muita Systeemitekniikan laitoksen prosesseja. Nykyinen kolonnin automaation toteutus vastaa enemmän sitä, kuinka automaatiojärjestelmän ja sen tietoliikenne toteutettaisiin todellisuudessa nykypäivänä [54]. Järjestelmää on helpompaa sekä turvallisempaa käyttää. Lisäksi se on havainnollistava opetuksen kannalta.

Kenttälaitteista uusittiin kolonnin akun pinnankorkeutta mittaava paine-erolähetin. Se korvattiin Endress+Hauserin Deltabar S PMD75 paine-ero lähettimellä. Kolonnin huipun painetta mittaava painelähetin korvattiin Endress+Hauserin Cerabar M PMP51 painelähettimellä. Molemmat lähettimet toimivat HART-protokollalla. Mittaussignaali voidaan lukea normaalina 4-20mA virtaviestinä. Digitaalisen HART-signaalin avulla voidaan konfiguroida lähettimiä sekä lukea diagnostiikkatietoja.

Syöttövirtauksen esilämmitystä ei pystytty aikaisemmin käyttämään esilämmityshöyryn säätöventtiilin väärästä mitoituksesta johtuen. Tislausprosessin ylösajo on ollut hidasta ja prosessissa tasapainotilojen saavuttaminen on vienyt aikaa. Esilämmityksen käyttämisen mahdollistamisen toivottiin nopeuttavan prosessia tai ainakin sen ylösajoa, joten venttiilin uusiminen nähtiin tarpeelliseksi. Kyseinen venttiili korvattiin Neles RAA-sarjan DN25 supistettuaukkoisella segmenttiventtiilillä. Venttiilipaketissa on mukana uusi jousipalautteinen pneumaattinen toimilaite. Venttiilin asennoittimeksi hankittiin digitaalinen Neles ND9103NP venttiilinohjain, joka kytketään jo olemassa olevaan Profibus PA-kenttäväylään. Profibus laitteet liittyvät prosessinohjaukseen aina Profibus DP väylän kautta, johon Profibus PA-väylä liitetään väylämuuntimen avulla.

Aiemmassa järjestelmässä kenttäväylän kautta liitettynä oli ainoastaan yksi laite. Päivityksen yhteydessä syötteen esilämmitysventtiilin ohjain liitettiin kenttäväylään. Toisaalta osa vanhoista virtaviestilaitteista haluttiin säilyttää, koska ne ovat täysin toimivia ja niitä on edelleen käytössä paljon teollisuudessa. Niiden avulla voidaan havainnoida automaatiolaitteiden toimintaa ja esimerkiksi venttiilin asennoittiminen viritystä.

Alitevirtauksen säätöön tarkoitetun venttiilin asennoittimen säädettävyys oli muuttunut huonoksi suuresta käyttöasteesta ja asennoittimen mekaanisesta virittämisestä johtuen. Uudistuksessa korvattavan esilämmityshöyryventtiilin asennoitin on ollut lähes käyttämättömänä, joten se vaihdetaan alitevirtauksen venttiilin asennoittimen paikalle.

Lauhdeveden lämpötilanmittaus oli ennen uudistusta viallinen. Rikkinäinen lauhdutusveden lämpötilaa mittaava lämpötila-anturi vaihdettiin. Uudeksi lämpötilamittaukseksi valittiin Endress+Hauserin Omnigrad TR-11 lämpötilamittaus. Kyseinen mittaus sisältää PT100-lämpötila-anturin sekä TM82 HART-lähettimen. Kyseisiä mittauksia hankittiin yhteensä viisi. Yli jäävät vanhat lämpötila-anturit sopivat varaosiksi ja niitä voidaan käyttää demonstraatiolaitteina opetustilanteissa.

Tarkka suunnitelma uusittavan instrumentoinnin sijoittamisesta prosessiin on tehty tämän diplomityön ohella tehdyssä opinnäytetyössä, jossa prosessin automaatioon liittyvää dokumentointia parannettiin [73].

6.3 Oppimisympäristön dokumentoinnin päivittäminen

Edellisen järjestelmänpäivityksen yhteydessä muutetut liitännät ja järjestelmän muutokset olivat dokumentoitu vajavaisesti. Tämä vaikeutti huomattavasti järjestelmän puutteiden ja uusintatarpeiden selvittämistä. Tässä diplomityössä tehdyn uudistuksen myötä pyrittiin parantamaan dokumentointia vastaamaan normaalin automaatiojärjestelmän toimitusprojektin mukaista dokumentointia. Lisäksi kiinnitettiin huomiota oppimisympäristöön liittyvään muuhun dokumentointiin kuten kirjanpitoon vikatilanteista.

Laitekaapin layoutsuunnittelu sekä ristikytkennän ja kenttäkotelon suunnittelu tehtiin opinnäytetyönä Tampereen Ammattikorkeakoululla. Lisäksi opinnäytetyössä piirrettiin automaatiojärjestelmän johdotuspiirikaaviot ja prosessin PI-kaavio uusiksi. Myös prosessiin liittyvistä automaatiolaitteista tehtiin erillinen listaus. [73]

Tilojen ja oppimisympäristöön liittyvien harjoitusten turvallisuusanalyysit tehtiin diplomityössä [74]. Kyseisen diplomityön tulokset liitetään osaksi oppimisympäristön dokumentointia,

Myös laboratorion kunnossapitoa ruvetaan dokumentoimaan. Laboratoriolle asetetaan vastuuhenkilö, joka tarkastaa laboratorion yleiset tilat sekä prosessin laitteet ennalta määritettynä aikana. Näin varmistetaan, että laboratorio ja sinne kuuluva laitteisto ovat toimintakunnossa tarvittavana aikana. Tarkastuksista ja huolloista pidetään kirjaa. Näin voidaan varmistaa, että voidaan ajoissa tilata esimerkiksi tarvittavat varaosat tai suorittaa huoltotoimenpiteet.

Tislausharjoitusten aikana on havaittu, että ulkotilan lämpötila vaikuttaa merkittävästi prosessin käyttäytymiseen. Lisäksi prosessi on monimutkainen ja vikaantuva. Tämän takia välillä voidaan joutua käyttämään esimerkiksi käsisäätöjä automaattisen säädön sijaan. Vuosien mittaan prosessissa tehtyjä harjoituksia on dokumentoitu. Nämä dokumentit säilytetään huolellisesti oppimisympäristön muun dokumentoinnin tapaan. Jatkossa oppimisympäristöä käyttävän opetushenkilöstö jatkaa tislausharjoitusten etenemisen dokumentointia. Käyttäjäkokemusten saaminen tämänkaltaisten prosessien tapauksessa on usein erittäin arvokasta onnistuneen prosessin säätötuloksen kannalta ja helpottaa mahdollisten ongelmatilanteiden selvittämisessä.

Oppimisympäristöön liittyvä dokumentaatio sisältää seuraavat asiat:

- Prosessin PI-kaavio
- Johdotuspiirikaaviot

- Laite- ja kenttäkaappien layoutkaaviot
- Laitteistolistaus
- Kenttälaitteiden manuaalit
- Huoltokirjanpito
- Tislattavan nesteen määrän kirjanpito
- Tislauspäiväkirja
- Höyrykattilan käyttöohjeet
- Automaatiojärjestelmän käyttöön liittyvät ohjeet
- Oppimisympäristön ja harjoitusten turvallisuusanalyysit
- Oppimisympäristön SysML-malli

Tislauskolonnijärjestelmään liittyvä dokumentointi parantuu merkittävästi entiseen nähden. Dokumentointi pyritään säilyttämään sähköisessä muodossa, jolloin se on yksinkertaisempaa saada käyttöön tarvittaessa. Lisäksi dokumentit ovat mahdollisuuksien mukaan muokattavissa formaateissa, joten muutoksien tekeminen on yksinkertaisempaa. Täsmällisempää ja monipuolisempaa dokumentointia voidaan käyttää hyväksi opetuksessa, oppimisympäristön ylläpidossa sekä mahdollisten tulevien päivitysten suorittamisessa.

6.4 Päivitetyn järjestelmän testaus

Automaatiojärjestelmän virtualisointi ja instrumentoinnin modernisointi uusiminen aiheuttaa systeemiin suurehkoja muutoksia ja vaatii testausta ennen käyttöönottoa. Prosessiasemat, kaapeloinnit ja kytkennät toimitetaan pääosin asennettuna ja käyttöönotettuna. Testausta varten määritettiin mallipohjaisen systeemisuunnittelun menetelmillä muodostetun systeemin mallin avulla testitapaukset, jotka kuvattiin aktiviteettikaavioissa.

Instrumentoinnin toimivuus täytyy testata uusien laitteiden asennuksen jälkeen. Valvomo-ohjelmasta tarkistetaan näyttääkö mittaukset oikeita lukemia. Näin voidaan varmistaa vaihdettujen mittausten toimivuus sekä kaikkien kytkentöjen toimivuus. Lämpötilamittausten testaus on syytä tehdä ennen käyttöä varsinaisessa prosessissa, vaikka lämpötilalähettimet ovat kalibroitu samalla asteikolle kuin entiset lähettimet. Ne voidaan testata huoneen lämpötilassa liitettynä valvomoon ja mittaamalla samalla lämpötilaa, jollain testatulla mittarilla tai vanhoilla testatuilla lämpötila-antureilla.

Akun pinnankorkeutta mittaava paine-erolähettimen mittauksen nollakohta ja maksimikohta täytyy asettaa kohdalleen. Akku ajetaan täyteen tislettä ja täytetään paineantureille kulkevat impulssiputket. Tämän jälkeen mittauksen arvot voidaan asettaa kohdalleen laiteohjelmiston avulla. Paine-erolähettimen mittaamaa pinnankorkeuden arvoa voidaan lisäksi verrata akun pinnakorkeutta näyttävään lasiputkeen.

Painelähettimen toiminta voidaan testata vertaamalla valvomosta saatavaa paineen arvoa kolonnin kyljessä olevaan analogiseen painemittarin arvoon. Tarvittaessa lähetin viritetään uudestaan.

Uuden venttiilin ja asennoittimen testaus suoritetaan asettamalla ohjaus valvomo-ohjelmistosta ja tarkastamalla kentällä venttiilin oikea asento. Venttiilipaketin vaikutus prosessiin täytyy testata oikean tislausprosessin koeajolla, jossa voidaan havainnoida kuinka esilämmityksen käyttö vaikuttaa prosessin ylösajoon ja itse prosessiin käyttäytymiseen.

Koko järjestelmä täytyy kuitenkin koeajaa ennen laboratorioharjoitusten suorittamista. Instrumentoinnin toiminnan testauksen jälkeen voidaan ajaa prosessia kylmänä eli kokeilla pumppujen toimivuutta ja muuttaa venttiilien ohjausta. Tämän jälkeen prosessia voidaan koeajaa kuumana ja varmistaa, että koko prosessi toimii kuten odotettu.

Virtualisoitua järjestelmää ja sen ominaisuuksia kokeillaan eri alustoilla. Ensin varmistetaan suunnitteluohjelmiston toimivuus sekä mikroluokasta, että kannettavalla laitteella. Suunnitteluharjoitusta valvova opettaja käy läpi mahdolliset muutokset suunnitteluohjelmistoissa ja tekee tarvittavat muutokset harjoitusmateriaaliin.

Tislausprosessiharjoituksessa on kerätty harjoituksen aikana prosessitietoa Matlab ohjelmistolla, jonka avulla on suoritettu simulointikokeita oikean prosessidatan pohjalta. Matlab käyttää prosessidatan keräykseen OPC-rajapintaa. Matlabin, Metso DNA-järjestelmän ja OPC:n yhteensopivuus täytyy varmistaa uudella alustalla ennen oikeiden prosessiharjoitusten alkua. Päivityksen yhteydessä hankittu Metso DNA info-palvelin tallentaa prosessitietoa ja dataa prosessin simulointiin sekä tarkasteluun voidaan tulevaisuudessa hankkia sen avulla.

Myös järjestelmän mahdollisia vikatilanteita on syytä kokeilla ennen varsinaista tutkimus- ja opetuskäyttöä. Tilanteita voivat olla esimerkiksi, kuinka järjestelmä toimii mahdollisessa tietoliikennekatkossa valvomoon tai miten järjestelmä suoriutuu sähkökatkosta.

Uuden systeemin toimintaa itse harjoituksissa voidaan arvioida lopullisesti vasta ensimmäisten harjoitusten jälkeen, mutta toimiva ja testattu prosessi sekä automaatiojärjestelmä mahdollistavat ainakin entisen järjestelmän tasoisen toiminnallisuuden. Seuraavassa luvussa on esitetty seikkoja mitkä parantavat oppimisympäristön toimintaa laboratorioharjoituksissa varsinaisen järjestelmäpäivityksen lisäksi.

6.5 Suunnitelma lisämuutoksista

Automaatiojärjestelmän ja laitteiden uusimisen lisäksi itse laboratoriotilat kaipaavat ehostusta ja tilojen selkeytystä. Tämä vaatimus tuli laboratoriossa järjestettävien kurssien opiskelijoiden palautteesta sekä opettajien haastattelusta. [72, 74]

Laboratoriotiloihin varastoidut sinne kuulumattomat ja tarpeettomat tavarat hävitetään tai varastoidaan muualle. Tiloihin hankitaan uudet tuolit, jotka on helppo kasata pieneen tilaan esimerkiksi päällekkäin. Automaatio-ohjelmiston virtualisointi mahdollistaa prosessin valvomisen ja suunnittelun esimerkiksi kannettavalta tietokoneelta. Tilassa aiemmin sijainneitta pöytäkoneita varten ei tarvita enää erillistä pöytätilaa. Tilaan voidaan hankkia myös helposti liikuteltavia ja muokattavia pieniä kirjoitus- tai pöytätasoja, jotta voidaan luopua isoista ja epäkäytännöllisistä pöydistä, jotka vievät paljon tilaa laboratoriossa.

Prosessin alakerta on tällä hetkellä eristetty muusta tilasta lukitulla verkkoaidalla tislattavan alkoholin säilytyksen takia. Verkkoaita voidaan rakentaa uudestaan pelkästään säiliöiden ympärille tai säiliöt voidaan varustaa lukittavilla hanoilla, jotta alakertakin saadaan vapaammin opetuskäyttöön.

Laboratoriohallissa, jossa tislauslaboratorio sijaitsee, on myös yliopiston muiden laitosten laboratoriotiloja. Laboratoriotila on siksi välillä meluisa. Myös itse tislausprosessin ajaminen aiheuttaa häiritsevää ääntä. Melun ongelma opetustilanteessa tuli esille opiskelijoiden palautteesta sekä opettajan kommenteista [72,74]. Erilaiset melueristeet auttaisivat laboratorion yläkerrassa kolonnin edessä. Eristeiksi riittäisivät esimerkiksi liikuteltavat väliseinät, jotka voidaan siirtää pois tarvittaessa, jos halutaan tarkastella lähemmin itse kolonnia. Myös ylä- ja alakerran välissä olevan ritilälattian kattaminen esimerkiksi pleksilasilla vähentäisi prosessitilan alakerrasta kuuluvaa meteliä tislausprosessin aikana.

Tislausharjoitusten aikana oppimisympäristössä on demonstroitu ja valvottu prosessia tietokoneruudulta tai kannettavalta videoprojektorilta heijastettavan kuvan avulla. Tilasta on kuitenkin puuttunut varsinainen pinta selvän kuvan heijastamiseksi. Nykyinen tietokoneruutu on auttamatta liian pieni tämänkokoiseen laboratoriotilaan ja tämä tuli esille opettajan ja opiskelijoiden palautteen perusteella [72,74]. Oppimisympäristön havainnointiin on välttämätöntä hankkia suurempi näyttötapa harjoituksia varten. Tilaan hankitaan suuri näyttö tai vaalea tasainen heijastuspinta tarkan kuvan saamiseksi videoprojektorin avulla.

Omatoimisia laboratorioharjoituksia varten laboratorion seinällä olevat prosessiin, laitteistoon ja tislauksen periaatteisiin liittyvät julisteet on syytä päivittää ajanmukaisiksi. Tämänhetkiset julisteet ovat kuluneita ja sisältävät vanhentunutta tietoa prosessilaitteista uudistuksen jälkeen. Vanhoja kenttälaitteita, jotka jäävät ylimääräisiksi uusittujen laitteiden tieltä voidaan käyttää esiteltäessä instrumentointitekniikkaa prosessienhallinnassa. Ne voidaan laittaa esimerkiksi esille itse laboratorioon.

Mahdollisia muita päivityksiä tislauskolonnin prosessiympäristöön tulevaisuudessa voi olla useampien laitteiden liittäminen kenttäväylään. Tämän uudistusprojektin aikana harkittiin signaalimuuntimien hankkimista standardivirtaviestein toimivien venttiilien

asennoittimien liittämiseksi kenttäväylään. Hankinta päätettiin jättää väliin, koska se ei sopinut budjettiin ja toteutuksen toimivuudesta ei ollut varmuutta. Kyseisellä ratkaisulla saataisiin kuitenkin havainnollistettua erilaista teknologiaa sekä sitä, kuinka vanhojen virtaviestilaitteiden integroiminen uudempaan kenttäväyläteknologiaan onnistuu.

Oppimisympäristöön hankittiin katkaistavat riviliittimet kenttäkoteloon ja ristikytkentään. Ajatuksena oli, että se mahdollistaa virtasignaalien mittaamisen suoraan johtimesta. Myös virtaviestin simulointi riviliittimen kautta valvomoon tai esimerkiksi venttiilille on uuden toteutuksen puolesta mahdollista. Näiden harjoitusten suorittaminen oppimisympäristössä vaatii yleismittarin ja laitteen, jolla pystytään simuloimaan 4-20 mA virtaviestejä. Tällaisia virtaviestiä simuloivia laitteita toimittaa esimerkiksi tunnettu elektronisten mittalaitteiden valmistaja Fluke.

Tällä hetkellä prosessissa virtausten mittaus tehdään pyörivään roottoriin perustuvien virtausantureiden avulla. Antureiden on havaittu likaantuvan, mikä vaikuttaa mittaustarkkuuteen. Laitteita täytyy puhdistaa usein. Virtausmittausten korvaamista kannattaa harkita tulevaisuudessa ratkaisuilla, joissa ei ole liikkuvia osia. Hyvänä vaihtoehtona olisivat coriolis-ilmiöön perustuvat virtausmittaukset. Tapa on erittäin tarkka ja sopii nesteiden ja kaasujen virtausten mittaamiseen. Mittalaitteet ovat helppoja huoltaa, koska niissä ei ole liikkuvia osia. Tämän myötä saataisiin myös uudenalaista instrumentointiteknologiaa opetuskäyttöön. Laitoksen prosessin virtausnopeudet ovat suhteellisen pieniä, mutta sen kaltaisiin virtauksiin on saatavilla coriolis-menetelmään perustuvia mittauslaitteita. Huonona puolena voinee mainita korkeat hankintakustannukset.

Laboratoriotiloihin muutoksia tehtäessä täytyy ottaa huomioon myös turvallisuus ja työviihtyvyysasiat. Oppimisympäristölle tehdään turvallisuusanalyysi Virtasen diplomityössä [74]. Kyseisen työn tulosten perusteella voidaan määrittää turvallisuusasiat, jotka täytyy ottaa huomioon omatoimisia laboratorioharjoituksia varten.

7. YHTEENVETO

Tässä luvussa esitellään, kuinka työssä käytetyt menetelmät sopivat prosessien hallinnan oppimisympäristön suunnitteluun sekä arvioidaan menetelmien sopivuutta erilaisten organisaatioiden käyttöön. Lisäksi arvioidaan, miten työssä onnistuttiin. Lopuksi esitetään ehdotuksia jatkotutkimukseen.

7.1 Johtopäätökset

Käytännönläheiset oppimisympäristöt ovat usein monimutkaisia kokonaisuuksia. Prosessien hallinnan ja automaation yhteiskäyttöinen oppimisympäristö sisältää useita sidosryhmiä, jotka suunnittelussa täytyy ottaa huomioon. Oppimisympäristö sisältää monipuolista laitteistoa, ohjelmistoa, teknistä tietoa sekä fyysisen ympäristön. Systeemisuunnittelun ja mallipohjaisen systeemisuunnittelun menetelmät soveltuvat tämän kaltaisten monimutkaisten systeemien suunnittelun apuvälineeksi.

Tässä työssä oppimisympäristön uudistuksen suunnittelussa käytettiin mallipohjaisen systeemisuunnittelun OOSEM-menetelmää, SysML-mallinnuskieltä sekä Magicdraw SysML-plugin suunnittelutyökalua. Mallinnusmenetelmä on erittäin laaja toimivat hyvänä apuvälineenä systeemiä suunniteltaessa. Olemassa olevasta oppimisympäristöstä muodostettiin menetelmän ja mallinnustyökalun avulla systeemin malli, jonka avulla havainnoitiin systeemin puutteita, säilytettäviä osia sekä parannuskohteita. Oppimisympäristössä suoritettavien harjoitusten perusteella muodostettiin käyttötapaukset, jotka toimivat toiminnallisina vaatimuksina. Käyttötapauksia mallinnettiin tarkemmin skenaarioanalyysin avulla aktiviteettikaavioiden muodossa. Lopullisen systeemin toteutus määritettiin niin, että skenaariot mahdollistuivat oikean systeemin avulla.

Vaatimusten määrittäminen ja jäljittävyyden ylläpitäminen olisi yksi tärkeimmistä seikoista, jossa mallipohjaista systeemisuunnittelua pystyi käyttämään hyväksi. Jäljitettävyyden ansiosta vaatimukset oli helppo kohdistaa systeemin eri komponenteille ja havainnoida niiden toteutuminen lopullisessa systeemissä suhteessa alkuperäisiin sidosryhmävaatimuksiin. Tislauskolonnin toiminnasta tehty aktiviteettikaavio helpotti ymmärrystä tislausprosessin periaatteesta sekä sen jakautumisesta systeemin eri komponenteille. Systeemin mallin osaksi tehdyt testitapaukset auttavat oppimisympäristön ja sen automaatiojärjestelmän testauksessa.

Tässä laajuudessa toteutetun projektin mallinnus onnistui käyttämällä SysML-kielen peruskirjastoja, mutta mitä erikoistuneimmaksi suunnittelukohteet menevät, sitä enem-

män vaaditaan ennalta määriteltyjä stereotyyppejä mallielementteinä. Tämä nopeuttaa ja helpottaa suunnitteluprosessia.

Vaikeutena mallipohjaisen systeemisuunnittelun käytössä oli suhteellisen jyrkkä oppimiskäyrä. Erikseen täytyi omaksua systeemisuunnittelun perusajatus ja erilaiset prosessit. Sen lisäksi täytyi hallita jokin mallipohjaisen systeemisuunnittelun menetelmä, käytettävä mallinnuskieli sekä lisäksi suunnittelutyökalun käyttö.

Mallipohjaisen systeemisuunnittelun edut nousevat opettelukustannuksia suuremmiksi, jos siitä tehdään osa jatkuvaa toimintaa. Pelkästään yhden tai ajoittaisen projektin takia mallipohjaisen systeemisuunnittelun käyttöönotto aiheuttaa turhan suuria vaikeuksia ja kustannuksia hyötyihinsä nähden.

7.2 Työn arviointi

Työn suoritus sujui suurin piirtein suunnitellussa aikataulussa. Vaikeuksia aiheutti tislausprosessin vajavainen dokumentointi, joten oppimisympäristön sekä tislausprosessin ominaisuuksien ja puutteiden selvittämiseen meni suunniteltua enemmän aikaa. Jopa aivan loppuvaiheessa ilmestyi uusia puutteita vajavaisen huoltodokumentoinnin takia, mutta ne onnistuttiin lopulta täyttämään projektin puitteissa. Työn perusteella suoritettu oppimisympäristön dokumentoinnin parantaminen edesauttaa mahdollisia huoltotoimenpiteitä tai uudistuksia tulevaisuudessa. Myös systeemisuunnittelun prosessien, mallipohjaisen systeemisuunnittelun menetelmien sekä SysML-kielen omaksuminen vaati suhteellisen paljon aikaa.

Kolonnin automaatiojärjestelmän päivitykseen tarvittava määrittely ja tilaus onnistuivat aikataulun rajoissa, mutta projektin edetessä sai havaita, kuinka pitkä prosessi automaatiojärjestelmän toteutus on. Projekteissa on monta eri osapuolta ja näiden välisten aikataulujen sovittaminen ei ole mutkatonta. Lisäksi olemassa olevien puutteellisesti dokumentoidun järjestelmän karakterisointi ja puutteiden havaitseminen on haasteellista.

7.3 Mahdollinen jatkotutkimus

Jo tämän työn aikana tulivat esille erilaiset yhteistyömahdollisuudet uusitun oppimisympäristön puitteissa sekä korkeakoulujen välillä. Tässä työssä uudistetun automaatiojärjestelmän dokumentointi tehtiin opinnäytetyönä Tampereen ammattikorkeakoululla.

Tampereen teknillisen yliopiston, Tampereen ammattikorkeakoulun sekä Tampereen yliopiston on suunniteltu yhdistävän toimintojaan tulevaisuudessa. Siksi tämänkaltainen oppimisympäristöjen yhteinen suunnittelu toimii hyvänä alkuna tulevaisuuden lisääntyvälle yhteistyölle ja mahdolliselle oppimisympäristöjen yhteiskäytölle.

Oppilaitosten välistä yhteistyötä voisi syventää ja opinnäytetöiden tekeminen samaan projektiin liittyen eri vaativuusasteilla voisi toimia muissakin yhteyksissä. Myös oppimisympäristöjä voitaisiin käyttää eri oppilaitosten välillä. Oppimisympäristöt ovat usein erikoistuneita ja niitä käytetään vain tiettyjen kurssien harjoituksiin. Tämä aiheuttaa tiloille ja laitteille alhaisen käyttöasteen. Opetustilojen keskinäinen yhteiskäyttö voisi tuoda kustannussäästöjä.

Lisäksi ajankohtaiseksi voisi tulla erilainen yritysyhteistyö oppimisympäristön yhteydessä. Oikea prosessi ja monipuolinen automaatiojärjestelmä voisi toimia ympäristönä erilaisille yrityskoulutuksille tai tutkimusprojekteille.

Tässä diplomityössä tehtyjen muutosten myötä oppimisympäristö mahdollistaa nyt myös monipuolisempien ja uudenlaisten opetustapahtumien järjestämisen. Tämän mahdollisuuden myötä kannattaa myös laboratorioharjoitusten kehittämiseen jatkossa panostaa.

LÄHTEET

[1] National research council. Miten opimme. Aivot, mieli, kokemus ja koulu. 1. Painos. Juva 2004. WS Bookwell Oy. 394 s.

[2] Opetushallitus. Perusopetuksen opetussuunnitelman perusteet 2004. Vammala 2004. Vammalan kirjapaino.

[3] Lindblom-Ylänne, S. Nevgi,A. Yliopisto- ja korkeakouluopettajan käsikirja. Vantaa 2003. WSOY.

[4] Toivonen, E. Opetus- ja tutkimuskäyttöön tarkoitettu tislauskolonni. Diplomityö. Tampere 1983. Tampereen teknillinen korkeakoulu, sähkötekniikan osasto. 98 s.

[5] Pullinen, R. Tislausprosessin automaation kehittäminen. Diplomityö. Tampere 1989. Tampereen teknillinen korkeakoulu, sähkötekniikan osasto. 86 s.

[6] Pullinen, R. Eräiden monimuuttujamenetelmien vertailu tislausprosessin säädössä. Lisensiaatintyö. Tampere 1993. Tampereen teknillinen korkeakoulu, sähkötekniikan osasto, säätötekniikan laitos. 52 s.

[7] Pitkäniemi, V. Sumean säädön sovellus tislauskolonnille. Diplomityö. Tampere 1995. Tampereen teknillinen korkeakoulu, sähkötekniikan osasto. 60 s.

[8] Pro-Suomi yhteishankinnat. WWW-sivu Saatavissa: http://prosessi.wikispaces.com/yhteishankinnat

[9] Keski-Uudenmaan ammattiopisto blogi. WWW-sivu. Saatavissa: http://blogi.keuda.fi/tislauskolonni-on-uudistettu-kemian-osastolla/

[10] Rask, O. Panosautomaation oppimisympäristö. Diplomityö. Tampere 2002. Tampereen teknillinen yliopisto.

[11] Laine, R. IDLE-oppimisympäristön suunnittelu ja toteuttaminen. Diplomityö. Tampere 2008. Tampereen teknillinen yliopisto. 59 s.

[12] Honkanen, J. Vesiprosessi oppimisympäristönä – Rakenteen ja ohjauksen suunnittelu. Opinnäytetyö. 2013. Satakunnan ammattikorkeakoulu, automaatioteknologia. 71 s.

[13] WWW-sivu. http://www.biolaakso.fi/kehitys/tutkimus-_ja_oppimisymparistot

[14] WWW-sivu. http://www.coursera.org/

[15] Niemistö, R. Ryhmän luovuus ja kehitysehdot. Helsinki, 2007. Palmenia.

[16] Kuusikorpi, M. Tulevaisuuden fyysinen oppimisympäristö. Käyttäjälähtöinen muunneltava ja joustava opetustila. Väitöskirja. Turku 2012. Turun yliopisto, kasvatustieteiden tiedekunta.

[17] Faulconbridge, R. Managing Complex Technical Projects: A Systems Engineering Approach. Norwood, MA, USA, 2002. Artech House. 271 p.

[18] Delligatti, L. SysML Distilled: A Brief Guide to the Systems Modeling language. USA 2013. Addison Wesley. 304 p.

[19] Rosenberg, D. Mancarella, S. Embedded Systems Development using SysML. OMG. Saatavissa: http://www.sparxsystems.com.au/resources/index.html

[20] Vasaiely, P. Interactive Simulation of SysML Models using Modelica. Bachelor's Thesis. Hamburg 2009. Hamburg University of Applied Sciences, Faculty of Engineering and Computer Science. 67 p.

[21] INCOSE. Systems engineering handbook - A Guide for System Life Cycle Processes and Activities. Version 3.2.2. October 2011.

[22] Buede, D. The Engineering Design of Systems. 2nd ed. New Jersey 2009. Wiley and Sons. 536 s.

[23] Weilkiens, T. Systems Engineering with SysML/UML: Modeling, Analysis, Design. Burlington, MA, 2008, Morgan Kaufman. 320 p.

[24] Schlager, K. Systems Engineering – key to modern development. Engineering Management , IRE – transactions 3(1956)3, pp 64-66.

[25] Kossiakof, A. Sweet, W. Seymor, S. Biemer, S. Systems Engineering Principles and Practice. 2nd ed. New Jersey 2011. Wiley and Sons. 560 p.

[26] Holt, J., Perry, S. SysML for Systems Engineering. IET. 2008. 352 p.

[27] Stevens, R., Brook, P., Jackson, K. Arnold, S. Systems Engineering: Coping with Complexity. 1st ed. England 1998. Prentice Hall. 392 p.

[28] Granholm, G., Karvonen I., Leino S-P., Viitaniemi, J., Salonen, T., Uoti, M., Alanen J. Katsaus kompleksisten järjestelmien elinkaaren suunnitteluun. Espoo 2013. VTT technology. 237 s. Saatavissa: http://www.vtt.fi/inf/pdf/technology/2013/T121.pdf

[29] Royce, W. Managing the Development of Large Software Systems. Proceedings, IEEE WESCON 1970.

[30] Haikala, I. Mikkonen, T. Ohjelmistotuotannon käytännöt. 12. Painos. Tampere 2011. Talentum Media Oy. 242 s.

[31] Boehm, B. A Spiral Model of Software development. Proceedings IEEE Second Sofware Process Workshop, ACM Software engineering note. 1986.

[32] Rook, P. Controlling software projects. Software engineering journal, 1(1986)1. pp. 7-16.

[33] Forsberg, K. Mooz, H. The Relationship of Systems Engineering to the Project Cycle.

[34] Forsberg K., Mooz H., Cotterman H. Visualizing Project Management: Models and Frameworks for Mastering Complex Systems. 3rd ed. NJ, USA, 2005. Wiley and Sons. 480 p.

[35] Hull, E., Jackson, K & Dick, J. Requirements Engineering. 3rd ed. Springer. London, 2011. 207 p.

[36] Gilb, T. Competitive engineering: A Handbook for Systems Engineering, Requirements Engineering and Software Engineering. Jordan Hill, Great Britain, 2005. Butterworth-Heinemann. 497 p.

[37] ISO/IEC 15288:2008. Systems and Software Engineering – System Life Cycle Processes. 71 p.

[38] Heumesser, N., Houdek, F. Towards Systematic Recycling of Systems Requirements. Proceedings of the 25th International Conference of Software Engineering. 2003.

[39] International Council of Systems Engineering Web page. Saatavissa: http://www.incose.org.

[40] Friedenthal, S., Moore, A., Steiner, R. A Practical Guide to SysML: The Systems modeling language. 2nd ed. 666 p.

[41] Fowler, M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Addison Wesley. USA 2004.

[42] Nolan, B., Brown, B., Balmelli, L., Bohn, T., Wahli, U. Model Driven Systems Development using Rational Products. 2008. IBM Red Books.

[43] Suomela, J-P. Ohjausjärjestelmän systems modeling language-pohjainen ohjelmistokehitys. Diplomityö. Tampere 2011. Tampereen teknillinen yliopisto, systeemitekniikan laitos. 134 s.

[44] OMG SysML-Modelica Transformation. Version 1.0. Saatavissa: http://www.omg.org/spec/index.htm

[45] Espinoza, H., Cancila, D., Selic, B., Gerard, S. Challenges in Combining SysML and MARTE for Model-Based Design of Embedded Systems. 2009.

[46] Saqui-Sannes, P., Hugues, J. Combining SysML and AADL for the Design, Validation and Implementation of Critical Systems. Toulouse, France, 2012.

[47] OMG Unified Modeling Language (OMG UML), Superstructure. Saatavissa: http://www.omg.org/spec/index.htm

[48] OMG MDA Guide version 1.0.1. Saatavissa: http://www.omg.org/spec/index.htm

[49] OMG Meta Object Facility (MOF) Core Specification version 2.4.2. April, 2014. Saatavissa: http://www.omg.org/spec/index.htm

[50] OMG Systems Modeling Language (OMG SysML™) version 1.3. 2012. Saatavissa: http://www.omg.org/spec/index.htm

[51] Cockburn, A. Writing effective use cases. Addison Wesley. 2001.

[52] Galloway, B., Hancke, P. G. Introduction to Industrial Control Networks. IEEE Communication Surveys & Tutorials vol 15 no 2. 2013.

[53] Powell, J., Vandelinde, H. Catching the Process Fieldbus: An Introduction to PROFIBUS for Process Automation. USA, 2013. Momentum Press. 176 p.

[54] Groover, M, P. Automation, Production Systems and Computer-Integrated Manufacturing. 3rd ed. Pearson Prentice-Hall. Upper Saddle River, NJ, USA, 2008.

[55] Huovinen, M. Large-scale Monitoring Applications in Process Industry. Väitöskirja. Tampere, 2010.

[56] Lehtonen, K. Älykkäät kenttälaitteet osana prosessin monitorointia. Diplomityö. Tampere 2008. Tampereen teknillinen yliopisto.

[57] Peltola, J. Kenttäväylien hyödyntäminen automaation toimitusprojekteissa. Diplomityö. Tampere, 2005.

[58] Liptak, B,G. Eren, H. Instrument Engineers' handbook, Volume 3: Process Software and Digital Networks. 4th ed. CRCPress.

[59] HART Foundation homepage. Saatavissa: http://en.hartcomm.org.

[60] Tooley, M. Plant and Process Engineering 360. 1st ed. Butterworth-Heinemann. 2009.

[61] Mackay, S., Wright, E., Reynders, D., Park, J. Practical Industrial Data Networks: Design, Installation and Troubleshooting. Elsevier 2004. 448 p.

[62] Chen, D., Nixon, M., Mok, A. WirelessHART: Real-Time Mesh Network for Industrial Automation. Springer. 2010.

[63] Deshpande, P. Distillation Dynamics and Control. New York 1985. Instrument Society of America. 499 p.

[64] Seppälä, A., Lampinen, M.J. Aineensiirto-oppi. Helsinki 2004. Otatieto. 219 s.

[65] Kiss, A., A. Advanced Distillation Technologies: Design, Control and Applications. April, 2013. Wiley. 414 p.

[66] OPC foundation homepage. Saatavissa: https://opcfoundation.org.

[67] Laitinen, O., Hännikäinen, H. ASE-1220 Systeemitekniikan perusteet 1: Kolonnidemo. Harjoitusmoniste.

[68] Laitinen, O., Jaatinen, A. ASE-2170 Automaatiojärjestelmät ja –suunnittelu: Tislauskolonnin automaatio. Laboratorioharjoitusohje. Maaliskuu 2012.

[69] Pietilä P., Laitinen O., Jaatinen A. ACI-31040 Automaatiolaitteet ja –verkot. Laboratorioharjoitusohje. Maaliskuu 2009.

[70] Pearce, P., Hause, M. ISO-15288, OOSEM and Model-Based Submarine Design.

[71] Karban, R., Andofalto P., Chiozzi G., Esselborn M., Schilling M., Schmid C., Sommer H., Zamparelli M. Model Based Systems Engineering for Astronomical Projects.

[72] Keskinen, S. Yhteiskäyttöisten laboratoriotilojen käyttötutkimus. Oheismateriaalit. Tampereen teknillinen yliopisto, Arkkitehtuurin laitos. Tampere, 2014

[73] Rajala, S. Opinnäytetyön pohjamateriaalit. Tampereen ammattikorkeakoulu. Tampere 2014.

[74] Virtanen, A. Diplomtyön oheismateriaali. Tampereen teknillinen yliopisto, Systeemitekniikan laitos. Tampere 2014.

[75] Roedler, G J. Jones C. Technical Measurement: A Collaborative Project of PSM, INCOSE, and Industry.

[76] Estefan, J., A. Survey of Model-Based Systems Engineering (MBSE) Methodologies. INCOSE. June 2008.

[77] Hochwalnner, M., Hörl, M., Dierneder, S. ,Scheidl, R. Some Aspects of SysML Application in the Reverse Engineering of Mechatronic Systems. Computer Aided Systems Theory – EUROCAST 2011. Revised Selected Papers, Part II. February 2011.

[78] Li, Q., Jiang, J., Rankin, D.,J. Evaluation of Delays Induced by Profibus PA Neworks. IEEE Transactions on Instrumentation and Measurement.